FEDERAL UNIVERSITY OF BAHIA

DOCTORATE DISSERTATION

A Fast Selective Grasping Algorithm with Deep Learning and Autonomous Dataset Creation on Point Cloud

Daniel Moura de Oliveira

Graduate Program In Electrical Engineering

Salvador
2024

DANIEL MOURA DE OLIVEIRA

A FAST SELECTIVE GRASPING ALGORITHM WITH DEEP LEARNING AND AUTONOMOUS DATASET CREATION ON POINT CLOUD

This doctorate dissertation was presented to the Graduate Program in Electrical Engineering of the Federal University of Bahia as part of the requirements for obtaining the degree of Doctor in Electrical Engineering.

Supervisor: Prof. Dr. André Gustavo Scolari Conceição

Salvador
2024

Ficha catalográfica elaborada pela Biblioteca Bernadete
Sinay Neves, Escola Politécnica - UFBA.

O48 Oliveira, Daniel Moura de.
A fast selective grasping algorithm with deep learning and autonomous dataset creation on point cloud/ Daniel Moura de Oliveira. – Salvador, 2024.
68f.: il. color.

Orientador: Prof. Dr. André Gustavo Scolari Conceição.

Tese (doutorado) – Programa de Pós-Graduação em Engenharia Elétrica, Escola Politécnica, Universidade Federal da Bahia, 2024.

1. Manipulador robótico. 2. Visão computacional. 3. Nuvem de pontos. 4. Preensão de objetos. 5. Algoritmo. I. Conceição, André Gustavo Scolari. II. Universidade Federal da Bahia. III. Título.

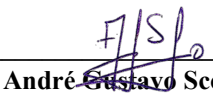
CDD: 629.892

A Fast Selective Grasping Algorithm with Deep Learning and Autonomous Dataset Creation on Point Cloud

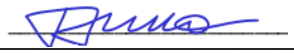
TESE DE DOUTORADO

Autor: Daniel Moura de Oliveira
Orientador: André Gustavo Scolari Conceição

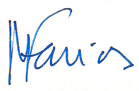
Tese de doutorado aprovada em 18 de Abril de 2024 pela banca examinadora composta pelos seguintes membros:



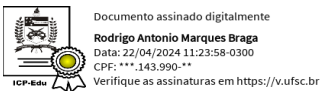
Prof. Dr. André Gustavo Scolari Conceição
Orientador - Universidade Federal da Bahia (UFBA)



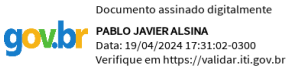
Prof. Dr. Tiago Trindade Ribeiro
Universidade Federal da Bahia (UFBA)



Prof. Dr. Paulo César Machado de Abreu Farias
Universidade Federal da Bahia (UFBA)



Prof. Dr. Rodrigo Antônio Marques Braga
Universidade Federal de Santa Catarina (UFSC)



Prof. Dr. Pablo Javier Alsina
Universidade Federal do Rio Grande do Norte (UFRN)

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to several individuals and companions who have played significant roles in the completion of this research.

First and foremost, I am deeply indebted to my thesis advisor Andre Scolari. Your unwavering support, invaluable guidance, and patience have been instrumental throughout this journey.

I extend my thanks to my family and friends for their support and understanding.

I also want to express my appreciation to my feline friend Paçoca, who, with her soothing presence and occasional distraction, made the long hours of research a little more enjoyable.

I want to acknowledge the support and resources provided by the academic community, without which this research would not have been possible. Lastly, this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

ABSTRACT

Grasping objects presents a multifaceted challenge influenced by variations in shape, perspective, material, roughness, and environmental conditions, making it a complex task. This research introduces an algorithm designed to address these challenges by employing point clouds since they allow a better notion of depth and geometry than RGB images. The proposed algorithm leverages geometric primitive estimation and lateral curvatures to identify optimal grasping regions swiftly and efficiently, where only the object geometry is used to analyze where to grasp.

To ensure the selection of desirable objects within the environment and to avoid undesirable ones, a purpose-built neural network, Point Encode Convolution (PEC), is introduced. PEC is tailored to utilize point clouds from RGB-D sensors and offers rapid execution and training times. The proposed design allows for efficient training and retraining, making it adaptable to diverse sets of objects. To expedite the training process, an autonomous dataset generation method is proposed. This method eliminates the need for manual annotation by autonomously generating data, and training is conducted within a simulation environment, such as Isaac Sim or any other simulation that allows object manipulation through scripts.

Validation of both algorithms, individually and in tandem, is conducted through the implementation of two grasp systems. The first system integrates the grasping algorithm with a neural network capable of object detection and 6D pose estimation. Initial validation occurs within the Webots and Gazebo simulations, where Gazebo was used for the visual validation and Webots for grasp validation due to its better physics handling without needing external plugins. However, due to certain limitations, the network component is excluded from subsequent experimental validation. The second system features the grasping algorithm with the neural network to be used on selective object grasping tasks.

Experimental validation is carried out using a UR5 robotic manipulator, an Intel RealSense D435 visual sensor, and a Robotiq 2F-140 gripper. The proposed neural network achieves a classification accuracy of 92.24% on a publicly available dataset. Meanwhile, the grasping algorithm attains an average success rate of 94% across all tested objects. The execution time of both algorithms is around 0.002 seconds each.

Keywords: Robotic Manipulator, Point Cloud, Deep Learning, Grasping, Computer Vision.

RESUMO

Preensão de objetos apresenta desafio de diversos tipos, devido a variação de forma, perspectiva, material, rugosidade e a condições do ambiente, fazendo da preensão uma tarefa complexa. Esse trabalho apresenta um algoritmo feito para lidar com esses problemas utilizando nuvem de pontos devido a sua melhor noção de profundidade e geometria do objeto em comparação a imagens RGB. O algoritmo proposto utiliza primitivas geométricas e curvaturas laterais para identificar o melhor local para se pegar um objeto de forma rápida e eficiente, onde somente a geometria do objeto é considerada para fazer a análise.

Para garantir a seleção de objetos de desejo para se realizar a preensão, uma rede neural de classificação, chamada de Point Encode Convolution (PEC), foi desenvolvida. A rede foi feita para ser utilizada em nuvem de pontos de sensor RGB-D e possui tempo baixo de execução e treinamento. Esse design flexível permite que a rede seja treinada e retreinada de forma eficiente, sendo facilmente adaptável para diversos grupos de objetos. Para auxiliar no processo de treinamento, um método de geração de datasets de forma autônoma foi proposto. Este método elimina a necessidade de anotação manual e é feito em um ambiente de simulação, como o Isaac Sim ou outro simulador que possua a opção de manipular objetos por script.

A validação de ambos os algoritmos, de forma individual e em conjunto, foi conduzida em dois sistemas. O primeiro sistema integra o algoritmo de preensão com uma rede neural capaz de detectar objetos e estimar a pose em 6D. A validação inicial foi feita em ambiente simulado do Gazebo e Webots, onde o Gazebo foi utilizado para validação visual e o Webots para validar a preensão devido a sua melhor física sem a necessidade de plugins externos. Entretanto, devido a certas limitações das redes de detecção de objetos e estimação de pose em 6D, a rede foi excluída na execução da validação experimental. Já o segundo sistema, combina ambos os algoritmos para execução de tarefas em preensão seletiva.

A validação experimental é conduzida usando um manipulador robótico UR5, o sensor visual RGB-D Intel Realsense D435 e uma garra Robotiq 2F-140. A rede neural proposta atingiu uma acurácia de 92.24% em um dataset de uso público, enquanto isso, o algoritmo de preensão atingiu uma média de 94% de sucesso em tarefas de preensão. O tempo de execução de ambos os algoritmos está por volta de 0.002 segundos cada.

Palavras-chave: Manipulador Robotico, Nuvem de Pontos, Aprendizado Profundo, Preensão, Visão computacional.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PUBLICATIONS

- de Oliveira, D.M.; Conceicao, A.G.S. A Fast 6DOF Visual Selective Grasping System Using Point Clouds. Machines 2023, 11, 540. https://doi.org/10.3390/machines 11050540.
- DE OLIVEIRA, DANIEL M. ; VITURINO, CAIO C. B. ; CONCEICAO, ANDRE G. S. . 6D Grasping Based On Lateral Curvatures and Geometric Primitives. In: 2021 Latin American Robotics Symposium (LARS). p. 138.
- DANIEL M. DE OLIVEIRA ; ANDRÉ G. S. CONCEIÇÃO . Preensão de objetos em 6D Utilizando Redes Neurais Convolucionais e Curvaturas. In: XV Simpósio Brasileiro de Automação Inteligente, 2021, Online, 2022.
- M. DE OLIVEIRA, DANIEL ; B. LEMOS, CÉZAR ; G. S. CONCEIÇÃO, ANDRÉ . Sistema de preensão robótica utilizando Redes Neurais Convolucionais e Primitivas Geométricas. In: Congresso Brasileiro de Automática 2020, 2020. Anais do Congresso Brasileiro de Automática 2020.
- Viturino, C. C. B. ; Santana, K. L. ; Oliveira, D. M. ; Lemos, C. B. ; Conceição, A. G. S. Redes Neurais Convolucionais para Identificação e Preensão Robótica de Objetos. In: XXIII Congresso Brasileiro de Automática, 2020, Santa Maria. XXIII Congresso Brasileiro de Automática, 2020.

LIST OF ACRONYMS

SSD	Single Shot MultiBox Detector
GPU	Graphical Processing Unit
PCL	Point Cloud Library
ROS	Robotic Operating System
CPU	Central Processing Unit
DOF	Degrees of Freedom
PEC	Point Encoder Convolution
FPS	Farthest Point Sampling
PCA	Principal Component Analysis
DH	Denavit-Hartenber
ADD	Average Distance
ICP	Iterative Closest Point
$EPnP$	Efficient Perspective-n-Poin
$VRAM$	Video Random-Access Memory
MLP	Multi-Layer Perceptron
VAE	Variational Auto Encoder
GGA	Generic Grasping Algorithm

LIST OF SYMBOLS

p	A point cloud
c	Object class
H	Homogenous transformation matrix
a	Link length
α	Link twist
d	Link offset
θ	Joint angle
T	Transformation
P	Origin of a frame
q	Joint Position
A	Constant coefficient
t	Time
v	Velocity
a_c	Acceleration
T_l	Translation
R	Rotation
$\mathbb{R}$	Real coordinate space
pi	pi constant
W_g	Gripper Stroke
H_g	Finger Width
H_o	Object height
K	Number of regions generated
f_p	PassThrough filter
λ	Eigen Value
$\vec{v}$	Eigen Vector
C	Covariance Matrix
p_c	Centroid of a Point Cloud Region
σ_{j*}	Curvature
L	Region Length
P	Grasping Score of a Region
φ	Rotation around x *(roll)* axis
θ	Rotation around y *(pitch)* axis
ψ	Rotation around z *(yaw)* axis

CHAPTER 1

INTRODUCTION

This introductory chapter will discuss the motivations for this dissertation research, the methodology, objectives, and the contributions to the literature.

1.1 MOTIVATION

Robotic manipulators have been used in the industry since the introduction of the first robotic manipulator, Unimate, in 1962. Unimate was used for automated diecasting at the General Motors plant in Ternstedt (MORAN, 2007). With the evolution of visual sensors, the use of robotic manipulators with computer vision to solve more complex tasks has become common in the literature and the industry, due to the flexibility it gives to the system since it can deal better with dynamic environments, where without the use of computer vision the environments would need to be mostly static. Over the last ten years, advancements in RGB-D sensors and computation power have improved tasks in diverse areas of robotics, such as robot navigation (Ferreira Neto et al., 2022), manufacturing operations (ARRAIS et al., 2019), and grasping (Carvalho de Souza et al., 2021).

Object detection from deep learning algorithms, like Single Shot MultiBox Detector (SSD) (LIU et al., 2016) and Yolov5 (QU et al., 2022), can be used to help detect objects in environments with multiple objects, making the visual system more robust and flexible when compared to traditional computer vision methods. However, since they use RGB images, they lack depth, resulting in a loss of precision in tasks that require accurate distance measurements or scale. To address this limitation, researchers have explored point cloud data in semantic segmentation tasks, such as PointNet++ (QI et al., 2017b), Semantic Point Cloud Segmentation Using Fast Deep Neural Network and DCRF (RAO et al., 2021), and EfficientLPS: Efficient LiDAR Panoptic Segmentation (SIROHI et al., 2022). While a point cloud contains 3D data (XYZ), it is still considered 2.5D because it suffers from occlusion. To overcome this limitation, multiple point clouds from different angles can be fused to generate a complete 3D representation (LIU et al., 2022).

In robotic manipulators, one common task is grasping or picking up objects. Visual systems aid in making these tasks more robust, as they allow the robot to perceive the desired object in relation to the camera and estimate the best region to grasp it whereas, without the use of computer vision, the object would have to always be in a previously known location, making it more susceptible to error. Works such as Grasp detection for assistive robotic manipulation (Jain; Argall, 2016) estimate the best place to grasp an object without any prior knowledge of the object using point clouds. Deep learning networks can also generate possible grasping strategies, as seen in the works of Mahler et al. (MAHLER et al., 2018) and Mousavian et al. (MOUSAVIAN; EPPNER; FOX, 2019). However, these approaches may have limitations, such as planar grasping, where

the gripper is always perpendicular to the table, or requiring heuristics to select one of the possible grasps returned by the network. To address these limitations, deep learning algorithms like PoseCNN (XIANG et al., 2017) and 6-Pack (WANG et al., 2020) have been developed. These algorithms detect the object, segment it, and estimate its pose using RGB and RGB-D images.

Reinforced learning and hierarchical learning approaches are currently at the forefront of robotic grasping research. Works such as Wang et al. (WANG et al., 2022) and Osa et al. (OSA; PETERS; NEUMANN, 2018) use these approaches to generate robust grasping deep learning algorithms. However, these types of algorithms are usually expensive to train due to the reinforced learning process and require robust hardware to run, making them difficult to retrain if necessary and challenging to use on low-cost hardware and systems.

This dissertation proposes a new selective grasping system that can handle diverse types of objects with low execution time. The system pairs a grasping algorithm with a classification deep learning network. Both algorithms are designed to have low hardware requirements for their use and will be validated in simulated and experimental environments.

1.2 METHODOLOGY AND OBJECTIVES

The goal of this dissertation is the development of a selective grasping system that has low execution time, low graphical memory usage and that can grasp unknown objects. The research is divided into two main areas: the development of a grasping algorithm capable of working with objects without prior knowledge, and the creation of a classification deep learning neural network to facilitate object selection and avoid grasping undesired objects in environments with multiple objects.

The first part of the research focuses on improving existing grasping algorithms. Building upon the work by Zapata et al. (ZAPATA-IMPATA et al., 2019), the proposed algorithm analyzes the entire object instead of solely focusing on the centroid. Additionally, it enhances geometric primitive estimation to detect objects that cannot be reduced to a primitive shape, improving the work of Jain et. al. (Jain; Argall, 2016). The validity of the grasping algorithm was established through simulations in Gazebo and further validation was conducted using the Webots simulator due to its better default physics when dealing with collisions.

The second part involves the development of a deep learning neural network for object classification using a point cloud from an RGB-D sensor. The proposed network features a simple architecture that enables its usage on more modest Graphic Processing Units(GPUs), like laptop GPUs, and swift retraining. To enhance the learning process and usability, a pre-processing algorithm is proposed. In conjunction with the deep learning network, a method for autonomously creating classification datasets is introduced. Validation of the proposed network was carried out using three datasets, including one publicly available dataset and two datasets generated using the proposed method.

Experimental validation of both algorithms was conducted using the UR5 robotic manipulator in conjunction with the Robotiq 2F-140 gripper. The Intel Realsense D435

RGB-D sensor was used for visual data acquisition. The algorithms were implemented using Python, with the support of libraries such as Numpy (HARRIS et al., 2020), Point Cloud Library (PCL) (RUSU; COUSINS, 2011), Open3D (ZHOU; PARK; KOLTUN, 2018), and Pytorch (PASZKE et al., 2019). The Robotic Operating System (ROS) was employed for controlling the robotic manipulator, and gripper, and acquiring data from the visual sensor. The proposed grasping algorithm and deep learning neural network can be accessed on GitHub: [1] and [2]. The system was validated using two hardware setups: a laptop equipped with an Intel i5 4200M Central Processing Unit (CPU) and Nvidia GT 850M GPU, and a desktop with a Ryzen 5 3600 CPU and Nvidia 3060ti GPU.

The main objectives of this research can be summarized as follows:

- Develop a grasping algorithm capable of efficiently grasping unknown objects using point clouds with low execution time.
- Create a deep learning neural network for object classification on point cloud, enabling selective grasping on low-power GPUs.
- Develop a method for easily generating classification datasets.
- Conduct comprehensive validation of both algorithms in both simulated and real-world environments.

1.3 CONTRIBUTIONS

The research conducted in this study has resulted in several contributions that have been published in scientific papers and presented at conferences. The key contributions and their corresponding publications are as follows:

- Validation of a grasping algorithm capable of efficiently grasping unknown objects with six Degrees of Freedom (DOF), which is an improvement of the works seen in Zapata et al. (ZAPATA-IMPATA et al., 2019) and Jain et al. (Jain; Argall, 2016). The initial version of the algorithm was validated using the Gazebo simulator and subsequently, the algorithm was further validated using the Webots simulator.
- The development of a deep neural network called Point Encoder Convolution (PEC) specifically designed for object classification on point clouds. This network is utilized by the proposed grasping algorithm to effectively select the desired object for grasping during experimental validation. Additionally, a preprocessing algorithm is introduced to enhance the learning process and generalization of the network by reorganizing the point cloud.
- The introduction of a method for the rapid creation of classification datasets using point clouds. This method utilizes a simulator and eliminates the need for manual labeling, enabling autonomous dataset generation. Paired with this method, a new dataset, named LARS Classification Dataset, is introduced.

[1] ⟨https://urlr.me/j2PyG⟩
[2] ⟨https://urlr.me/xBPZv⟩

- Thorough experimental validation of the proposed grasping algorithm, the PEC network, and the dataset creation method using a robotic manipulator UR5, a Robotiq 2F140 gripper, and an Intel Realsense D435 RGB-D sensor.

1.4 DISSERTATION STRUCTURE

Previously, this Chapter presented the motivations, methodology, objectives, and contributions of this research.

Chapter 2 provides an overview of related work and the current state-of-the-art literature published in recent years.

Chapter 3 introduces two proposed grasping systems: one for simulation experiments and another for experimental validation. The rationale behind the use of two different systems is also discussed in this chapter.

Chapter 4 focuses on the development of the deep learning network called PEC. The chapter presents the network's architecture, the preprocessing algorithm employed, the method for creating classification datasets, and the results obtained from three datasets, including one public dataset and two datasets generated using the proposed method for dataset creation.

Chapter 5 elaborates on the proposed grasping algorithm, detailing each step of the grasping process and highlighting its improvements over existing algorithms in the literature. The chapter also showcases the simulated and experimental results for both the grasping algorithm in isolation and the selective grasping system incorporating the PEC network.

Finally, in Chapter 6, the dissertation concludes by summarizing the entire discussion, reiterating the contributions made, and presenting the overall results achieved.

CHAPTER 2

RELATED WORK

This chapter examines relevant scientific literature, including research papers, theses, dissertations, and books that have provided inspiration and conceptual foundations for the present study. The purpose of this literature review is to contextualize and inform the research, elucidating key concepts and methodologies utilized to achieve the study's objectives.

Section 2.1 focuses on exploring and evaluating state-of-the-art algorithms in the existing literature, specifically those pertaining to 6D object pose estimation and object detection.

Section 2.2 will illuminate an algorithm utilizing neural networks specifically designed for point cloud processing. Given the research's emphasis on point cloud-based grasping, this algorithm's examination is particularly relevant, as it provides insights into leveraging machine learning techniques to enhance object perception.

Lastly, Section 2.3 will provide an in-depth exploration of the definition and theory of robotic grasping, complemented by a survey of seminal works in the literature that have significantly contributed to shaping and inspiring the research's approach to grasping tasks.

2.1 6D POSE ESTIMATION AND OBJECT DETECTION

In robotics, object detection and pose estimation play crucial roles in various applications, including mobile robotics and robotic manipulators on tasks that require finding yourself in an environment or getting the correct object pose to grasp. These tasks are essential for identifying objects in the environment and are particularly beneficial for localization and grasping tasks. With the advancements in neural networks, they have become a prominent tool in the field. One such work, PoseCNN (XIANG et al., 2017), presents a convolutional neural network that accomplishes two tasks: object segmentation, utilizing another work by the same author, DA-RNN (XIANG; FOX, 2017), and 6D pose estimation. The pose estimation is achieved through two modules: one for translation and the other for rotation. The architecture of PoseCNN is depicted in Figure 2.1, where we can see that it has three interconnected stages to generate labels for each object and uses those labels with another output of the network to estimate the 6D pose of each object.

To complement the neural network, the authors also introduced the YCB dataset, which contains 21 objects, as seen in Figure 2.2. The dataset comprises 92 videos with 133,827 frames, and each object has an associated 3D model available.

The pose estimation results are evaluated based on the Average Distance (ADD) metric, where the pose estimation is deemed inaccurate if the estimated pose error is greater than 10%. The authors justify this threshold, stating that errors above 10% could lead to failed grasping attempts. The ADD metric is defined as follows, where

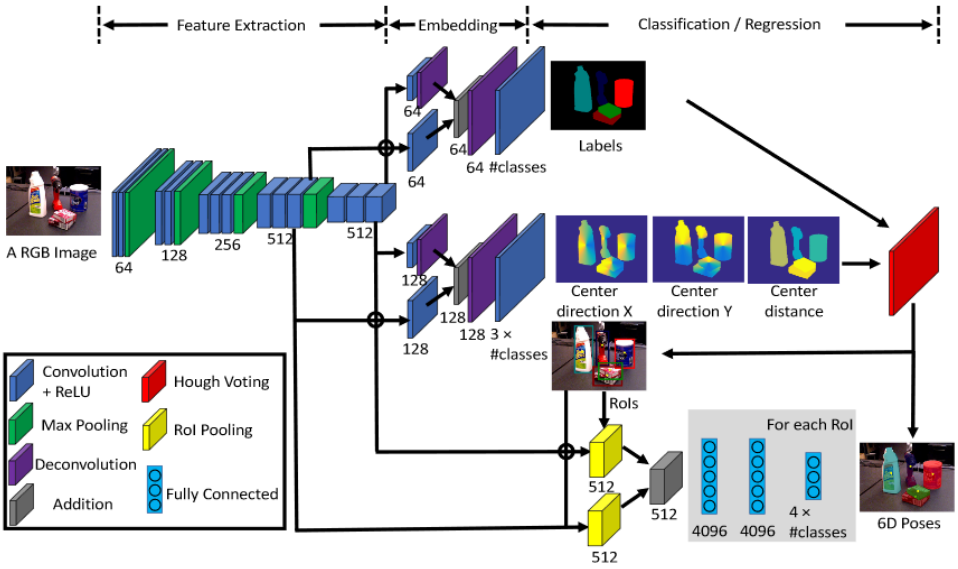


Figure 2.1: PoseCNN's architecture. Source: (XIANG et al., 2017).

Figure 2.2: Objects of the *YCB dataset*. Source:(XIANG et al., 2017).

R and T_l represent the rotation and translation of the object, $\hat{R}$ and $\hat{T}_l$ denote the estimated rotation and translation from the network, m represents the number of points that represents the object, and M is the number of points from the 3D model of the object, where the used dataset provides the 3D model of each object to estimate this metric:

$$ADD = \frac{1}{m} \sum_{x \in M} \| (Rx + T_l) - (\hat{R}x + \hat{T}_l) \| \tag{2.1}$$

In scenarios involving symmetric objects, the ambiguity of points can pose challenges.

The ADD-S metric is utilized to address this issue, which calculates distances using the closest points. The ADD-S can be defined as follows:

$$ADD - S = \frac{1}{m} \sum_{x_1 \in M} \min_{x_2 \in M} \| (Rx_1 + T_l) - (\hat{R}x_2 + \hat{T}_l) \| \tag{2.2}$$

The evaluation of the results is performed on both RGB and RGB-D images. In the case of RGB images, PoseCNN achieved an ADD of 53.7% and an ADD-S of 75.9%. In contrast, when utilizing RGB-D images, the performance significantly improved, resulting in an ADD of 79.3% and an ADD-S of 93.0%. The notable enhancement in results when using RGB-D images underscores the importance of incorporating depth information for accurate pose estimation.

In contrast to PoseCNN, Densefusion (WANG et al., 2019) proposes a neural network architecture for estimating the 6D pose of an object by fusing RGB and RGB-D images to generate a point cloud for pose estimation. The architecture of Densefusion can be seen in Figure 2.3, where segmentation is accomplished using DA-RNN, similar to PoseCNN, where the network work by using a Convolutional Neural Network (CNN) and a PointNet to do color and geometric embeddings to estimate the object pose. However, instead of using Iterative Closest Point (ICP) for pose refinement, Densefusion employs another neural network dedicated to refining the pose, as depicted in Figure 2.4.

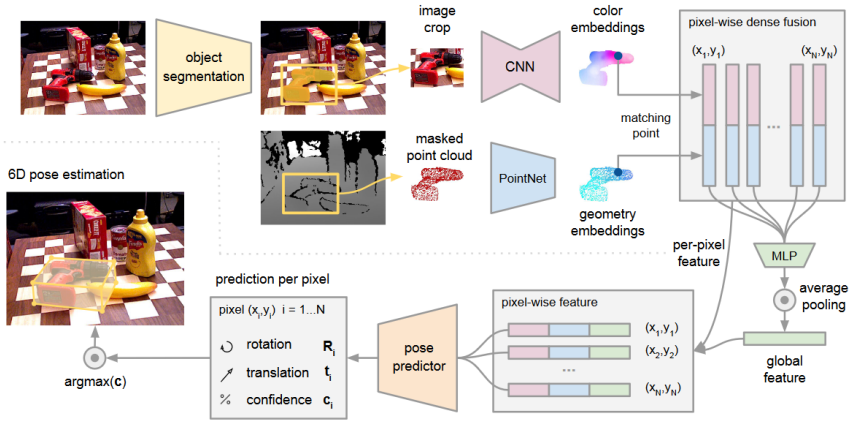


Figure 2.3: Densefusion architecture that uses semantic segmentation mixed with a convolutional neural network to generate color embeddings and a PointNet to generate geometric embeddings to estimate the object pose. Source:(WANG et al., 2019).

During validation, Densefusion utilizes the ADD-S metric, but with a different threshold for considering an error as wrong. It sets the threshold to any error above 2cm, which the author deems as the ideal maximum error for grasping tasks. As a result, Densefusion achieves an ADD-S score of 96.8%, an improvement over PoseCNN's 93.2% when using the same metric. Comparing the performance of both algorithms, Densefusion demonstrates its capability to estimate the object's pose in a mere 0.06 seconds. Within this time, 0.03 seconds are used for semantic segmentation, 0.02 seconds for pose estimation,

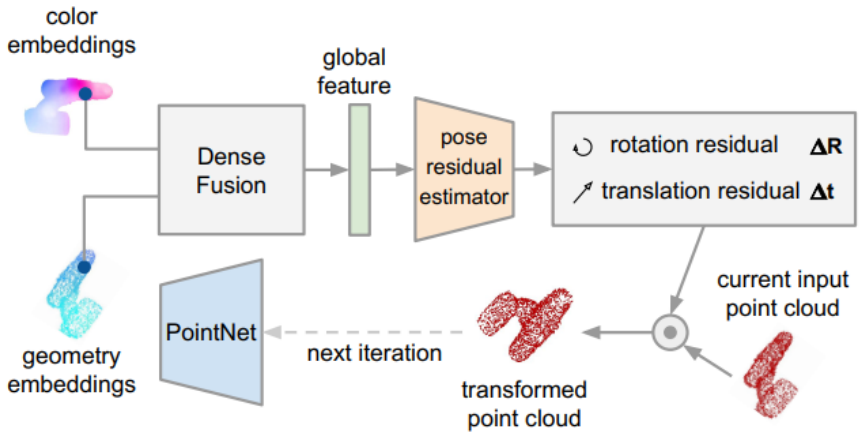


Figure 2.4: Pose refinement architecture of the DenseFusion that uses a neural network to refine the pose instead of using ICP like PoseCNN. Source:(WANG et al., 2019).

and 0.01 seconds for pose refinement. In contrast, PoseCNN requires 0.17 seconds for pose estimation and a longer 10.6 seconds for pose refinement. The faster and more accurate performance of Densefusion makes it a compelling choice for object pose estimation, and its ability to handle RGB-D data demonstrates the potential benefits of combining multiple sources of information in robotic perception tasks.

6D Object Pose Regression via Supervised Learning on Point Clouds (GAO et al., 2020) utilizes a similar approach to Densefusion by fusing RGB and RGB-D images to create a point cloud, which is then used to estimate the object's pose. Similarly, it follows a modular approach to estimate translation and rotation, akin to PoseCNN. Figure 2.5 depicts the network architecture, where it uses two BaseNets, the convolution architecture proposed by Qi et al. (QI et al., 2017b), to estimate the translation and rotation separately while using a depth image and the objects' semantic segmentation as input. A distinguishing feature of this network is that it not only predicts the pose but also reconstructs the point cloud representation of the object in the estimated pose. This attribute enhances its usability in grasping tasks, as it provides a complete 3D model unaffected by issues like occlusion or missing points. During validation, the network achieved an ADD-S of 94.7%, ADD of 82.7%, and 90.3% of the poses had an error below 1cm.

In contrast, Segmentation-driven 6D Object Pose Estimation (HU et al., 2019) adopts a different approach compared to other networks discussed in this section. Its architecture, as shown in Figure 2.6, primarily focuses on semantic segmentation and feature extraction. The pose estimation is performed using Efficient Perspective-n-Point (EPnP) (LEPETIT; MORENO-NOGUER; FUA, 2009), which compares the features from the RGB image with those of the object's 3D model. While validating on the YCB dataset, this method yielded the lowest result, an ADD of 39%. However, it exhibited an impressive execution time of 22FPS, surpassing other methods and achieving real-time performance.

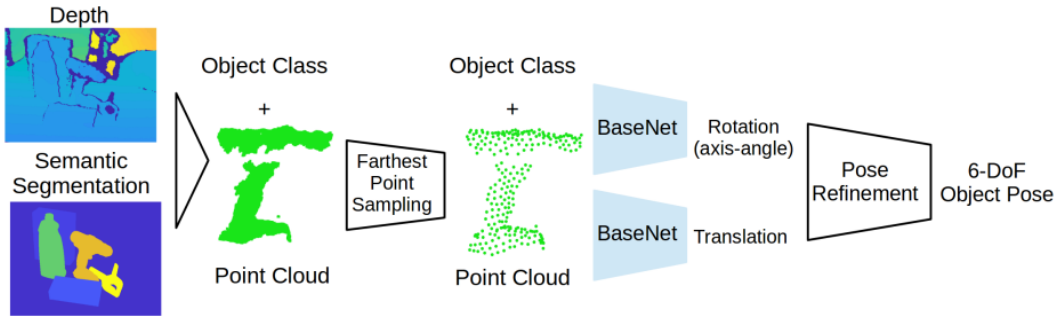


Figure 2.5: 6D Object Pose Regression via Supervised Learning on Point Clouds architecture. Source:(GAO et al., 2020).

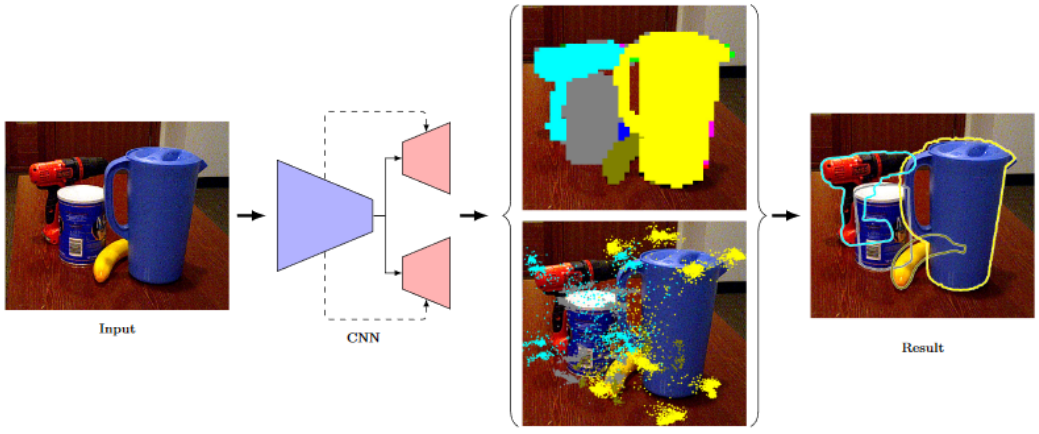


Figure 2.6: Segmentation-driven 6D Object Pose Estimation architecture. It uses three convolution neural networks to segment the objects and estimate features. Source:(HU et al., 2019).

During testing in a simulated environment, several limitations surfaced that could impede experimental validation. Firstly, the networks only worked with identical objects bearing the same texture. Objects that were similar but not identical resulted in false positives. Secondly, a considerable number of false positives were encountered, with only PoseCNN and Segmentation-driven 6D Object Pose Estimation delivering acceptable results in simulation. Lastly, the networks could only estimate one object instance. The second and third issues are visualized in Figure 2.7. Among the problems identified, the first one was particularly critical for altering the proposed grasping system during experimental validation. Extensive efforts would be required to generate enough data using objects available in the laboratory since these networks necessitate labeled data for both pose estimation and semantic segmentation. For simulated validation, Segmentation-driven 6D Object Pose Estimation was chosen due to its superior performance and the least occurrence of false positives. PoseCNN was omitted due to its high hardware re-

quirements for pose estimation, demanding a minimum of 6GB of Video Random-Access Memory (VRAM).

Figure 2.7: Single instance object detection and pose estimation with false positive object detection.

2.2 NEURAL NETWORKS ON POINT CLOUDS

The application of point clouds in robotics has witnessed a notable surge in recent years. This rise is particularly evident in the realm of autonomous vehicles, where research utilizing point clouds has gained significant traction. Among the works that harness point clouds for tasks like semantic segmentation, PointNet (QI et al., 2017a) stands as a pivotal contribution. Its architectonic design is versatile and applicable for both classification and semantic segmentation tasks. Notably, this architecture has served as a foundational framework for various other deep learning endeavors involving point clouds, including tasks like pose estimation. Densefusion (WANG et al., 2019), for instance, capitalizes on PointNet as a component of its pose estimation and refinement approach. Figure 2.8 portrays the PointNet architecture in detail, where we can see that the proposed architecture is divided into two stages: one for classification and another for semantic segmentation that uses the classification output.

Another creation by the author of PointNet, PointNet++ (QI et al., 2017b), takes PointNet's architecture and augments it with grouping and sampling operations, enhancing its precision for classification and segmentation duties. Figure 2.9 elucidates the structure of this advanced deep learning model, where it improves the previous PointNet architecture by adding a preprocessing step and multiple PointNet to generate the final output. When evaluated on the *Model40 Shape Classification* dataset, which contains

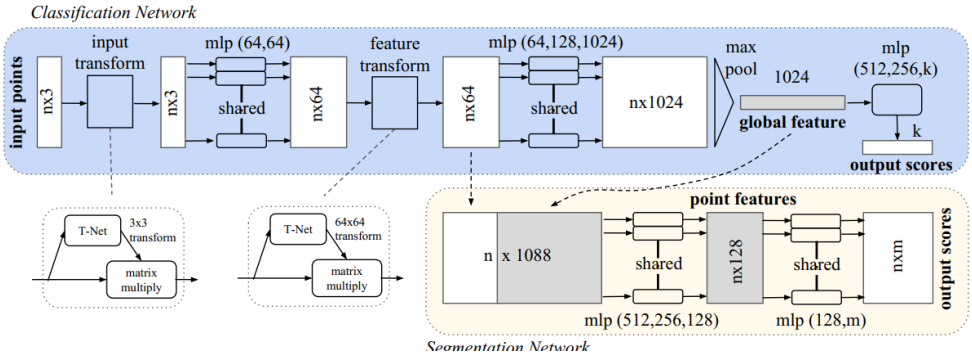


Figure 2.8: The PointNet architecture. Source: (QI et al., 2017a).

3D objects classified into 40 classes, each having multiple samples, PointNet achieved an accuracy of 89.2%, while PointNet++ achieved a superior 91.9% accuracy.

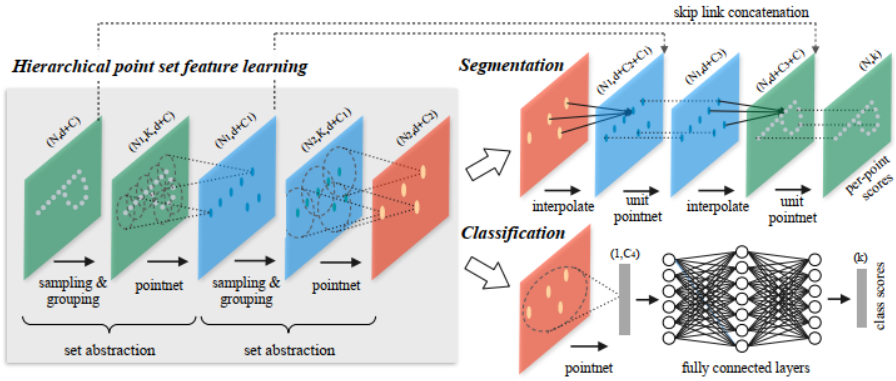


Figure 2.9: The PointNet++ architecture. Source: (QI et al., 2017b).

A novel convolutional architecture, PointCNN (LI et al., 2018), emerges as an alternative approach for processing point clouds within deep neural networks. This architecture introduces the concept of *X-Conv*, a specialized convolution mechanism tailored for the irregular and unordered data inherent in point clouds. Illustrated in Figure 2.10, *X-Conv* amalgamates matrix operations, Multi-Layer Perceptron (MLP), and convolution. Figure 2.11 shows the PointCNN full proposes architecture for each task, where for a classification task it suggests using two *X-Conv* with a single classification output, for multi-class classification it proposes the use of two *X-Conv* but with multiple output classes, and for semantic segmentation the use of four *X-Conv* layers. Notably, when assessed against the *Model40 Shape Classification* dataset, PointCNN exhibited a slight enhancement in accuracy, achieving 92.5%, in comparison to PointNet++.

Diverging from the preceding networks, CurveNet (XIANG et al., 2021) introduces a distinctive approach by incorporating curves and shape estimation from point clouds for

ALGORITHM 1: $\mathcal{X}$-Conv Operator

Input : $\mathbf{K}, p, \mathbf{P}, \mathbf{F}$
Output : $\mathbf{F}_p$ ▷ Features "projected", or "aggregated", into representative point p

1: $\mathbf{P}' \leftarrow \mathbf{P} - p$ ▷ Move $\mathbf{P}$ to local coordinate system of p
2: $\mathbf{F}_\delta \leftarrow MLP_\delta(\mathbf{P}')$ ▷ **Individually** lift each point into C_δ dimensional space
3: $\mathbf{F}_* \leftarrow [\mathbf{F}_\delta, \mathbf{F}]$ ▷ Concatenate $\mathbf{F}_\delta$ and $\mathbf{F}$, $\mathbf{F}_*$ is a $K \times (C_\delta + C_1)$ matrix
4: $\mathcal{X} \leftarrow MLP(\mathbf{P}')$ ▷ Learn the $K \times K$ $\mathcal{X}$-transformation matrix
5: $\mathbf{F}_\mathcal{X} \leftarrow \mathcal{X} \times \mathbf{F}_*$ ▷ Weight and permute $\mathbf{F}_*$ with the learnt $\mathcal{X}$
6: $\mathbf{F}_p \leftarrow \text{Conv}(\mathbf{K}, \mathbf{F}_\mathcal{X})$ ▷ Finally, typical convolution between $\mathbf{K}$ and $\mathbf{F}_\mathcal{X}$

Figure 2.10: X-Conv architecture. Source: (LI et al., 2018).

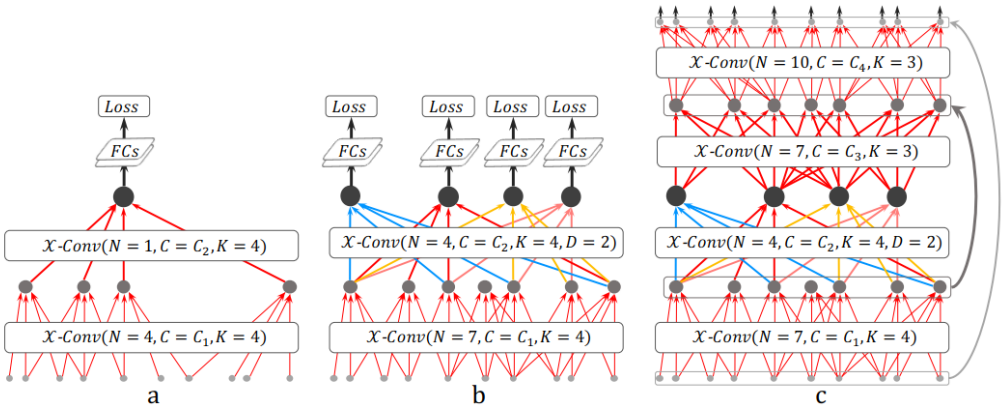


Figure 2.11: PointCNN architecture where (a) is the architecture for classification, (b) for multi-class classification, and (c) semantic segmentation. Source: (LI et al., 2018).

object classification and segmentation. Figure 2.12 shows the CurveNet architecture and its complexity, where it has three stages to generate the final output using a mix of Multi-Layer Perception(MLP), points grouping, concatenations, pooling, and other operations proposed by the author. Leveraging its complexity, CurveNet achieved superior outcomes compared to both PointNet++ and PointCNN, boasting an impressive accuracy of 94.2% on the *Model40 Shape Classification* dataset.

2.3 ROBOTIC GRASPING

According to Robot Grasping Foundations (BRIOT S.; KHALIL, 2015), a grasp is commonly defined as a set of contacts on the surface of the object, whose purpose is to constrain the potential movements of the object in the event of external disturbances. The primary purpose of a grasp is to constrain the potential movements of the object, particularly in response to external disturbances. In the context of robotic manipulators, grasping involves utilizing a set of contact points, often facilitated by a gripper, to securely hold an object. This process enables precise control over the object and is essential for various tasks, such as pick-and-place operations.

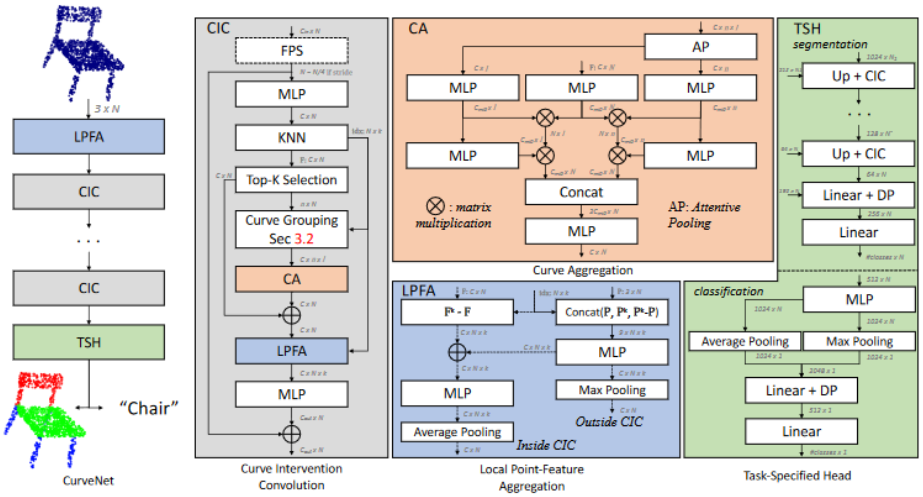


Figure 2.12: CurveNet architecture. Source: (XIANG et al., 2021).

In the work Grasp Detection for Assistive Robotic Manipulation (Jain; Argall, 2016), a grasping algorithm based on geometric primitives is introduced. This algorithm relies on geometric primitives and is designed for deployment in conjunction with an RGB-D visual sensor and a robotic manipulator equipped with a three-finger gripper. To determine the most suitable grasp for an object, the author leverages curvature information derived from the object's surface. By analyzing the estimated geometry, the algorithm categorizes the object into three distinct geometric shapes: cube, cylinder, or sphere. Figure 2.13 shows a grasp estimation using this work, where it can be seen that it suggests two places to grasp the object, from its front and the top.

However, it's worth noting that this approach encounters limitations when dealing with more complex objects possessing intricate geometries. In cases where an object's shape consists of multiple intricate components, attempting to approximate it as a single simplified geometry can lead to inaccurate grasp planning and potentially result in unsuccessful grasping attempts.

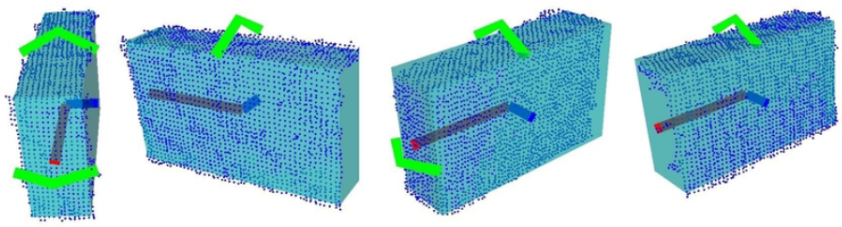


Figure 2.13: Grasp example from Grasp Detection for Assistive Robotic Manipulation, where the green represents the gripper position. Source: (Jain; Argall, 2016).

In High Precision Grasp Pose Detection in Dense Clutter (GUALTIERI et al., 2016), an approach to grasp pose detection is presented, which combines analytical algorithms with deep learning techniques for the classification of grasping poses. Notably, the proposed algorithm is specifically designed for grippers with two fingers, and the author does not recommend its utilization with other types of grippers. The key methodology involves generating potential grasping points through random sampling of points, followed by the estimation of their surface normals. Subsequently, a feasibility check algorithm is employed to validate whether a grasp of those regions is viable. Once deemed feasible, a classification network is invoked to categorize each grasp and determine the optimal grasp configuration. An illustration of the grasp estimation process using this algorithm is depicted in Figure 2.14, where it shows the possible grasps generated, represented by the yellow gripper.

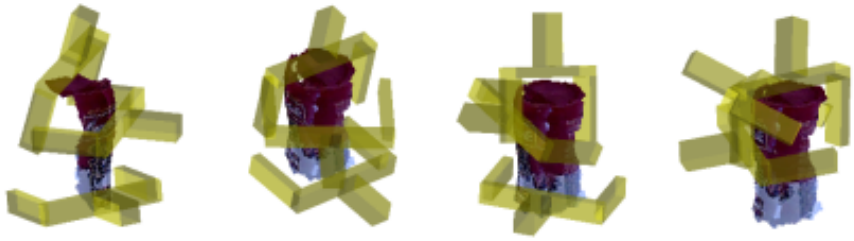


Figure 2.14: Possible grasp regions estimated by the work High precision grasp pose detection in dense clutter. Source: (GUALTIERI et al., 2016).

On a related note, Fast Geometry-Based Computation of Grasping Points On Three-Dimensional Point Clouds (ZAPATA-IMPATA et al., 2019) introduces an alternative grasping algorithm tailored for industrial robotic manipulators. Distinguished by its versatility, this algorithm is capable of handling diverse objects without prior knowledge of their specific shapes or forms. The methodology revolves around analyzing points in proximity to the object's centroid. This involves estimating the curvature around each point, determining their orientation using PCA, assessing the plane distance on which they are situated, and considering the gripper configuration. However, a notable limitation of this approach is its emphasis on searching around the object's centroid, potentially posing challenges when dealing with intricate objects where the centroid region might not yield optimal grasping outcomes. Figure 2.15 shows how the proposed grasp algorithm works, where it selects a pair of points on the lateral extremity of the object and generates a point ranking based on multiple factors defined by the author.

GraspNet (MOUSAVIAN; EPPNER; FOX, 2019) stands as an advanced grasping algorithm harnessing the capabilities of deep learning to ascertain optimal grasp configurations for objects. While the neural network adeptly classifies potential grasps, an auxiliary heuristic remains essential to pinpoint the ultimate choice. Central to its architecture is the utilization of a Variational Auto Encoder (VAE), where both the encoder and decoder find their foundation in the PointNet++ framework. The training process

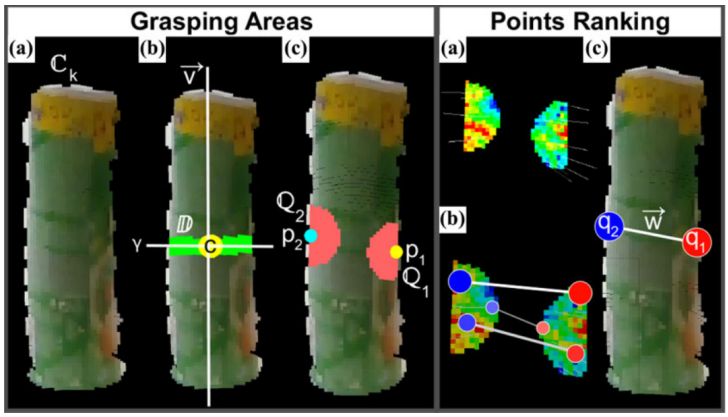


Figure 2.15: Grasping estimation process for Zapata et. al. work. Source: (ZAPATA-IMPATA et al., 2019).

relies on reinforcement learning, wherein the neural network evolves through a trial-and-error approach within a simulation environment. Notably, this simulation encompassed an extensive 10,816,720 potential grasp scenarios, yielding a mere 20% success rate. From this voluminous dataset, training data was judiciously partitioned, with 30% designated for successful grasps and the remaining 70% for instances of grasping failure. Empirical assessments on diverse objects culminated in a remarkable 88% success ratio for the executed grasps.

Hierarchical Policies For Cluttered Scene Grasping With Latent Plan (WANG et al., 2022) introduces an innovative 6D grasping algorithm characterized by its ability to predict optimal grasp locations and their corresponding trajectories. Figure 2.16 provides a visual depiction of its architecture, which adeptly harnesses the PointNet and PointNet++ frameworks to estimate the local and global trajectory of the manipulator. The network leverages semantic segmentation to partition objects within the visual sensor's field of view, subsequently generating viable grasp trajectories and forecasting potential collisions. Mirroring GraspNet, this algorithm employs reinforcement learning and relies on simulation-based training. Notably, a unique feature of the approach is its hierarchical nature, enabling a portion of the training to be conducted on offline data to expedite the learning process.

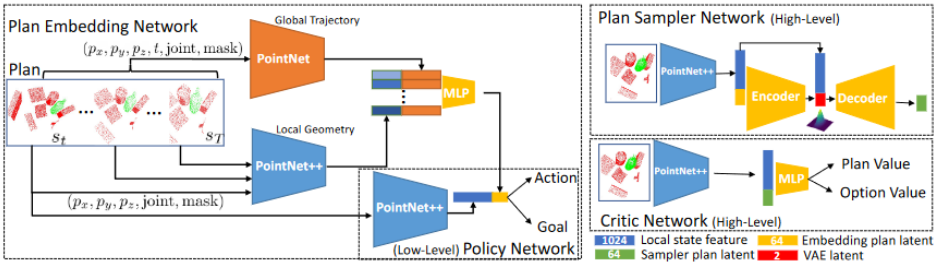


Figure 2.16: Network architecture proposed by Wang et al.. work. Source: (WANG et al., 2022).

CHAPTER 3

PROPOSED GRASPING SYSTEMS

This chapter introduces two proposed grasping systems: one designed for a simulated environment, using a neural network to estimate the 6D pose for the proposed grasping algorithm, and another for experimental validation, using a traditional method to estimate the 6D pose for the grasping algorithm. Initially, the grasping system was intended to utilize a neural network for simultaneous pose estimation and semantic segmentation. However, during experimental tests, it became apparent that the performance of existing networks was not satisfactory, since the tested neural network only worked with the same objects it was trained and would require extensive retraining using objects available in the laboratory. Generating a dataset for this purpose proved to be challenging, as it necessitated manual annotation of poses for each frame and semantic segmentation.

To address this limitation, an alternative approach was adopted. The proposed grasping system was modified to utilize alternative techniques for pose estimation and object segmentation that were better suited for the experimental setup. These techniques offered improved performance and eliminated the need for labor-intensive dataset generation and retraining.

3.1 SIMULATED SYSTEM

The first proposed system was utilized for simulated experiments, as depicted in Figure 3.1. This system was used in the published works: Oliveira et al. 2021a (OLIVEIRA; CONCEICAO, 2021) and Oliveira et al. 2021b (OLIVEIRA; VITURINO; CONCEICAO, 2021).The system operates according to the following steps:

- Initially, an RGB image is employed to perform object segmentation and estimate their poses using a neural network to detect and estimate the 6D pose.
- Using the provided object 3D model by the dataset used to train the neural network and the estimated pose, the object is integrated into the point cloud with the mimicked pose. This approach effectively bypasses issues such as occlusion or defects in the point cloud since it uses a 3D model of the object instead of the point cloud from the sensor. Poisson Disk Sampling (YUKSEL, 2015) is a technique used to convert a 3D model into a point cloud in a way that ensures the generated points are evenly distributed while avoiding overcrowding.
- The proposed grasping algorithm is applied to determine the optimal grasping region.
- Based on the 6D pose of the object and the identified optimal grasping region, the robotic manipulator is instructed to grasp the object.

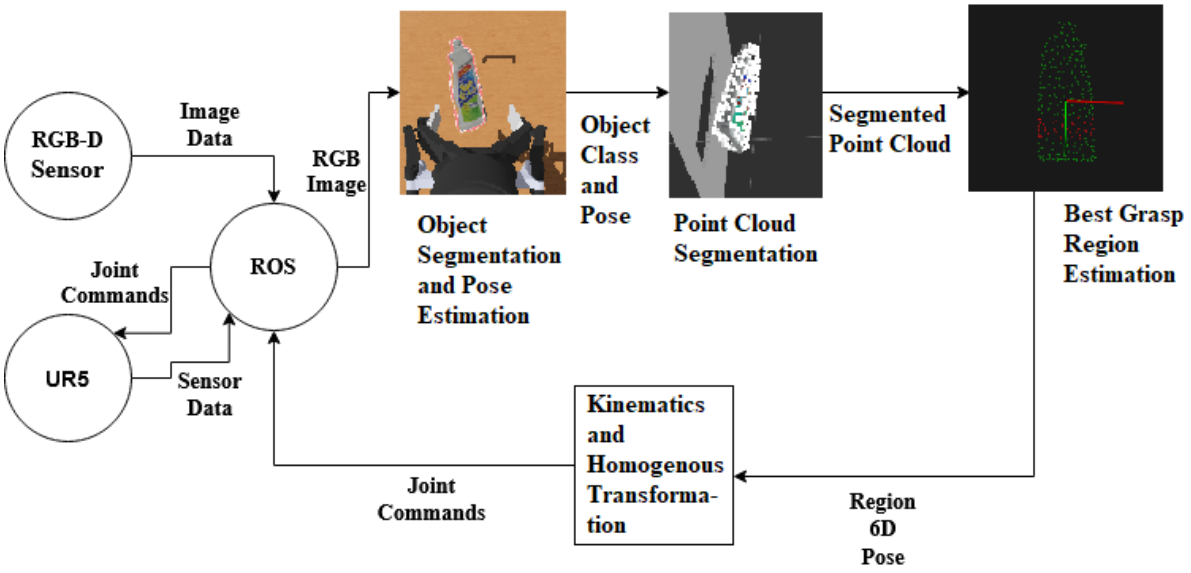


Figure 3.1: Proposed grasping system for simulated validation.

Although this system exhibited promising results in simulated environments, it encountered challenges when transitioning to experimental validation, including:

- False positives: The tested deep learning networks for pose estimation and semantic segmentation produced a significant number of false positives.
- Limited capability to estimate multiple objects of the same class: The networks could only estimate the pose of one object per class, disregarding additional instances of the same object in the scene.
- Restricted applicability to specific objects: The deep learning networks only functioned effectively with the same objects they were trained on, failing to perform well with similar objects or those possessing different color patterns.

Addressing these limitations through extensive retraining would require extensive laborious manual labeling for semantic segmentation and pose estimation. Consequently, the system was adapted to employ alternative techniques for pose estimation, object segmentation, and classification in order to overcome these challenges.

3.2 EXPERIMENTAL SYSTEM

This system was used for experimental validation with the objective of overcoming the limitations mentioned previously. This system was used in the published work by Oliveira et al. 2023 (OLIVEIRA; CONCEICAO, 2023). The grasping system overview proposed in this research can be seen in Figure 3.2, where the stages of the system are as follows:

- Filtering stage:

1. Input point cloud: the initial point cloud p_i returned by the RGB-D sensor.
2. Gripper and table removal: the point cloud after being filtered by removing the gripper and the table, having only the objects over the table. The filtering process works as follows:
   (a) Removing the gripper by limiting the z-axis (depth axis) between $[z_{min}, z_{max}]$ meters, where z_{min} and z_{max} are the parameters of the system setup. The gripper and objects far away from the camera are removed, for example, the floor.
   (b) Removing the table using plane segmentation (RUSU; COUSINS, 2011) and an Outliner filter.
3. Clustering to separate the objects in the point cloud. Returns to the point cloud p_j of each object, with $j = 1, \ldots, n$, where n is the number of objects.

- Classification stage:

4. Preprocess the data by normalization, downsampling using Farthest Point Sampling (FPS) (QI et al., 2017b), and grouping the points. Returns the point cloud p_{fj}, with $j = 1, \ldots, n$, where n is the number of objects.
5. The objects' point cloud is classified using the proposed classification network PEC. Returns the objects class c_j, with $j = 1, \ldots, n$, where n is the number of objects.
6. Select the object with the class desired c to be grasped, so we can select only the object we want to grasp and avoid undesirable objects and return its point cloud p to the grasping algorithm. From this step on, selective grasping is possible, as only the point cloud of the desired object will be considered for the grasping stage.

- Grasping stage:

7. Receive the object's point cloud p and estimate its 6D pose using Principal Component Analysis (PCA).
8. Estimate the grasping region of the selected object in the camera's field of view. The proposed algorithm uses lateral curvatures and geometric primitives to estimate the grasping region.
9. Returns the 6D pose to grasp the desired object.

To overcome the limitations mentioned earlier, several changes were implemented in the segmentation, pose estimation, and object classification components compared to the simulated system. These modifications aimed to enhance the system's performance and address the challenges faced during experimental validation.

For segmentation, a Plane Segmentation (RUSU; COUSINS, 2011) technique was employed. In this approach, the objects are assumed to be placed on a plane, such as a table, and by removing the plane, only the objects remain. While this technique

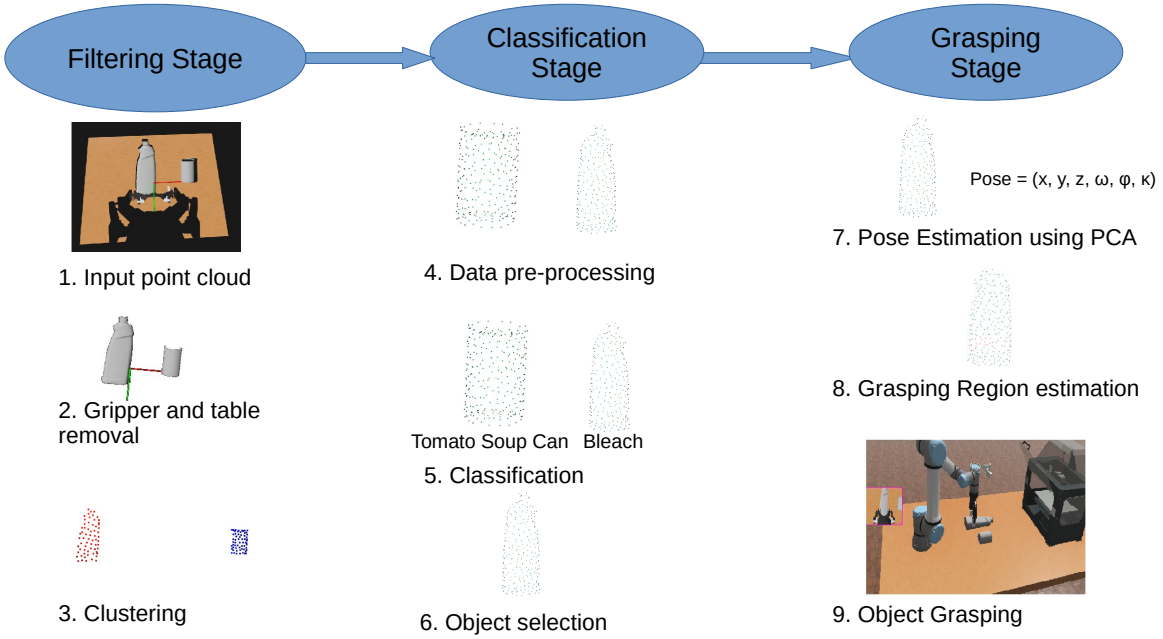


Figure 3.2: Proposed grasping system for experimental validation.

introduces the limitation that objects can only be segmented when placed on planes, it allows for the validation of the grasping system with a wide range of objects.

In terms of pose estimation, the PCA method was adopted, as suggested by Zapata et al. (ZAPATA-IMPATA et al., 2019) and Jain et al. (Jain; Argall, 2016).

To enable selective grasping, a dedicated deep learning network was developed solely for object classification. This approach was chosen as classification networks are easier to train and require less complex dataset creation compared to semantic segmentation datasets.

These changes in segmentation, pose estimation, and object classification components collectively enhance the system's capabilities and enable effective grasping in real-world scenarios.

CHAPTER 4

POINT ENCODER CONVOLUTION

This chapter focuses on introducing the proposed deep learning neural network, Point Encoder Convolution (PEC), which is designed specifically for object classification on point clouds acquired from RGB-D sensors. The chapter provides insights into the following aspects of the PEC network:

- Data Preprocessing: Section 4.1 presents the data preprocessing flow for training and classification. This involves preparing the point cloud data to ensure its compatibility with the network and optimize classification performance.
- Network Architecture: Section 4.2 details the architecture of the PEC network, highlighting its design choices and components that enable accurate and efficient object classification. It also explains the network's structure, layers, and parameters.
- Proposed Datasets and Creation Method: Section 4.3 discusses the datasets utilized for training and evaluating the PEC network. These datasets are specifically designed to facilitate object classification tasks on point clouds. Also, a novel method for creating classification datasets without manual labeling is introduced. The Section explains how this method autonomously generates labeled datasets, removing the labor-intensive process of manual annotation. This approach streamlines dataset creation and enables faster experimentation and training.
- Results: The results obtained from the evaluation of the proposed deep learning network, PEC, are presented in Section 4.4. The evaluation encompasses three datasets: two datasets created using the proposed dataset creation method and one publicly available dataset. The section provides a detailed analysis of the network's performance and its ability to accurately classify objects across these datasets.

4.1 DATA PREPROCESSING

Prior to applying the data to the PEC network for object classification, a preprocessing step is performed to enhance the learning and generalization process. Given a group of point clouds $p_j \subset \mathbb{R}^3$ as input, where $j = 1, \ldots, n$ and n is the number of objects, the centroid p_c of each point cloud p is defined as the average of all points p_i within the point cloud p, where $i = 1, \ldots, N$ and N is the total number of points:

$$p_c = \frac{1}{N} \sum_{i=1}^{N} p_i, \quad p_i \in p. \tag{4.1}$$

Using the object centroid, the point cloud of the object is translated to the origin frame by subtracting the centroid from each point in the point cloud, resulting in the translated point cloud p_f:

$$p_f = p - p_c. \tag{4.2}$$

This normalization process centers the data around the centroid, preventing overfitting during the training step and improving generalization when classifying objects. Subsequently, the object is sampled using Farthest Point Sampling (MOENNING; DODGSON, 2003), which reduces the number of points in the point cloud to a fixed number required as input for the classification network.

To improve the learning process, the point cloud will be reorganized by creating a pseudo-organized point cloud. This grouping and reorganization improved the results, as will be shown in the results later in this section, and is necessary because a point cloud is stored in a single 1D array, and, different from an RGB image, the order of the points on the array has no pattern and can impact learning and generalization. The pseudo-organized point cloud is created by following these steps, where the initial point cloud will remain intact but the point cloud array will be reorganized:

- Select the first point of the point cloud array p_inital and apply a k-d tree (BENTLEY, 1975) to identify i points near the target point, where i must be divisible by the total number of points in the point cloud p_f
- Group those points together and remove them from the original point cloud.
- Repeat this process until all points have been grouped.
- Once all points are grouped, calculate the centroid of each group using Equation 4.1, and sort the groups based on their distance to the origin, as proposed by Qi et al. (QI et al., 2017b).
- Merge all the groups to reconstruct the point cloud p_f with the points reorganized.

By performing these preprocessing steps, the input data is appropriately prepared for classification, facilitating the learning process and enabling more accurate object classification using the PEC network.

4.2 NETWORK ARCHITECTURE

Figure 4.1 illustrates the complete architecture of PEC, with an input size of 64 points and 5 classes, resulting in a total of 890K parameters. The architecture is composed of an autoencoder architecture, incorporating 1D convolutions, 1D transpose convolutions, MLP, and a bottleneck in the middle, a common architecture employed in deep learning work due to its efficient. Each convolution layer includes a batch normalization step to enhance training stability and consistency. The parameterization encompasses the configuration of convolution layers (input size, output size, kernel size, stride, padding), linear layers (input size, output size), and transposed convolution layers (input size, output size,

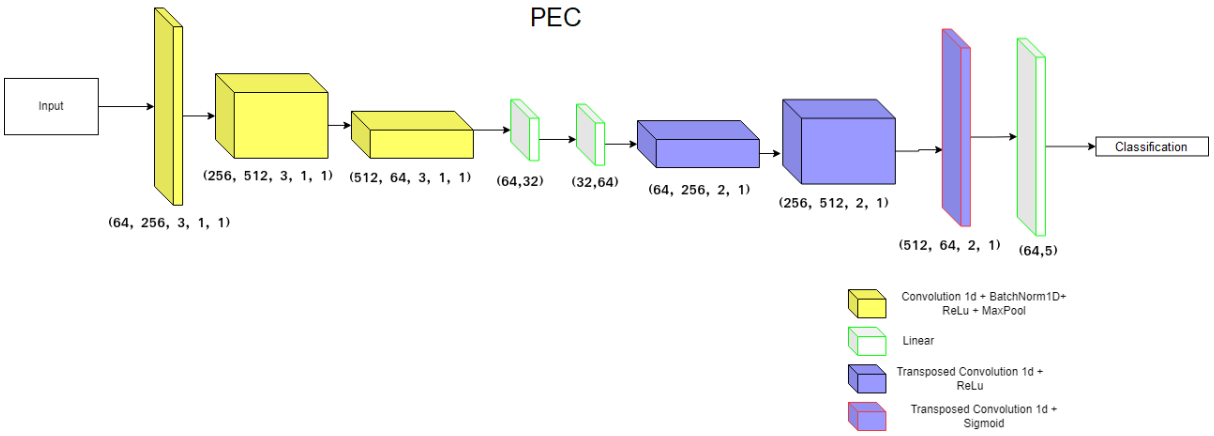


Figure 4.1: PEC's network architecture.

kernel size, stride). For loss calculation, the cross-entropy loss function is employed due to its superior generalization performance when classifying objects that differ from those encountered during training. The Adam stochastic optimization method (KINGMA; BA, 2014) is chosen for its faster and more effective training compared to other optimization algorithms. The network outputs a class c and a confidence score, obtained by applying the softmax function (GOODFELLOW; BENGIO; COURVILLE, 2016) to the network's output. The network's hyperparameters are empirically determined.

4.3 PROPOSED DATASETS AND CREATION METHOD

To validate the proposed network, three datasets are utilized: ModelNet10 (WU et al., 2015), a publicly available dataset commonly used in the literature, and two datasets created using a novel method. The proposed method for dataset creation involves the use of a simulator. In this research, Isaac Sim (MONTEIRO et al., 2019) is employed due to its script manipulation capabilities, visual fidelity, and ability to generate random object poses while capturing point cloud images at each frame. The support for script allows us to spawn and manipulate objects using Python script, which allows us to easily know what object is on the scene and generate random poses to generate annotations. This approach ensures that each class is covered from multiple angles and regions, generating a large number of samples. Figure 4.2 illustrates the simulator interface and the dataset generation process and Figure 4.3 shows the diagram with the process to generate the dataset, where first we spawn an object of known class and then generate random poses of the object and save them each frame. The process is finished when the desired number of samples is generated.

For the created datasets, each consists of 5 classes. One dataset contains two objects representing each class, while the other dataset features only one object per class. Random poses in $\mathbb{R}^3$ space are generated to create one thousand samples for each object, with orientations ranging from $[-\pi, \pi]$ and positions ranging from $[-2$ m, 2 m$]$ where simulated RGB-D sensors capture point clouds for each pose. Figure 4.4 showcases the first dataset created using the proposed method, which includes objects adapted from

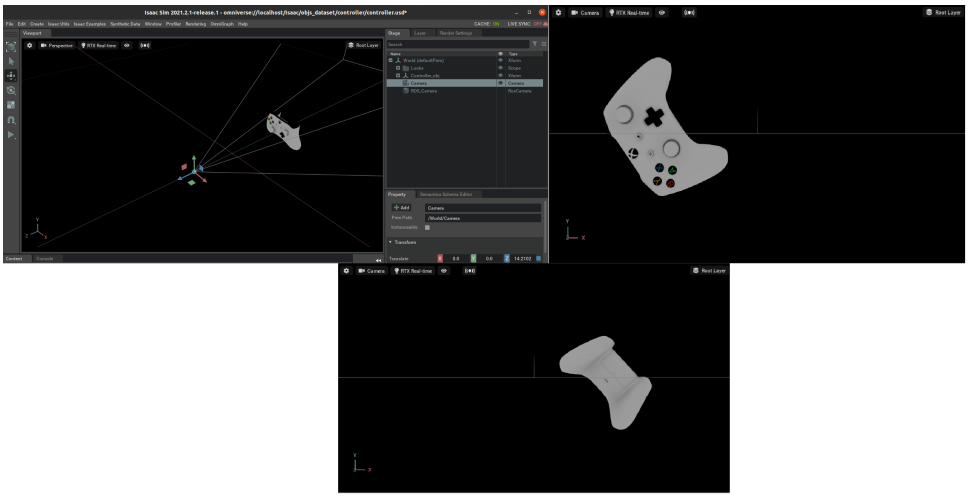


Figure 4.2: Dataset generation through Isaac Sim.

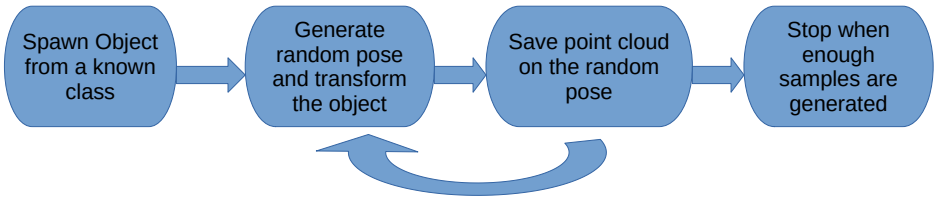
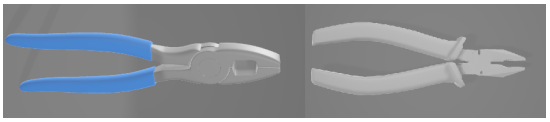


Figure 4.3: Diagram of the process to generate a dataset.

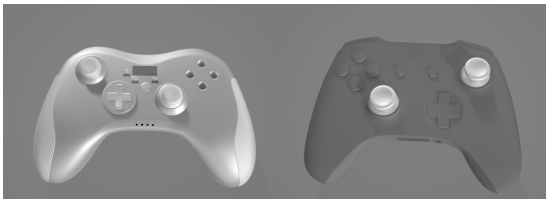
the YCB Dataset (XIANG et al., 2017). The second dataset, named the LARS Classification Dataset, incorporates objects similar to those found in the laboratory, specifically selected for experimental validation. Figure 4.5 presents the objects and their respective classes in the LARS Classification Dataset.

One challenge of this approach is that the simulated data is ideal and does not accurately represent the defects typically present in real point clouds. To address this and improve generalization in real-world experiments, a random subset of points is removed from a randomly selected region within each data sample. The number of points removed varies between 0% and 40% of the total points. This amount of points was chosen to avoid deleting too many points and simulate defects and other problems that could happen with an RGB-D sensor in real life. This technique does not affect the training accuracy but

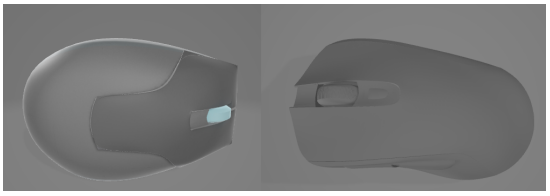
Figure 4.4: Objects of the YCB dataset adapted for classification.



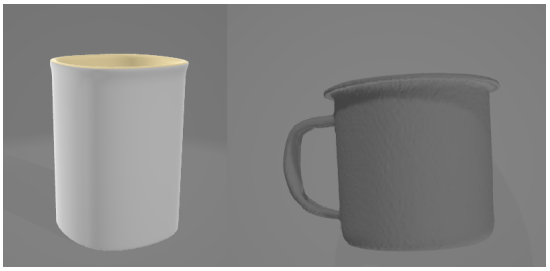
(a) Objects of the class pliers.



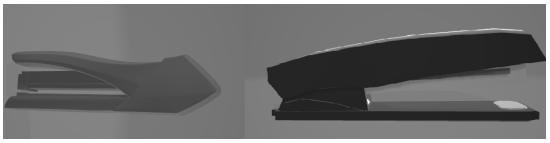
(b) Objects of the class joypad.



(c) Objects of the class mouse.

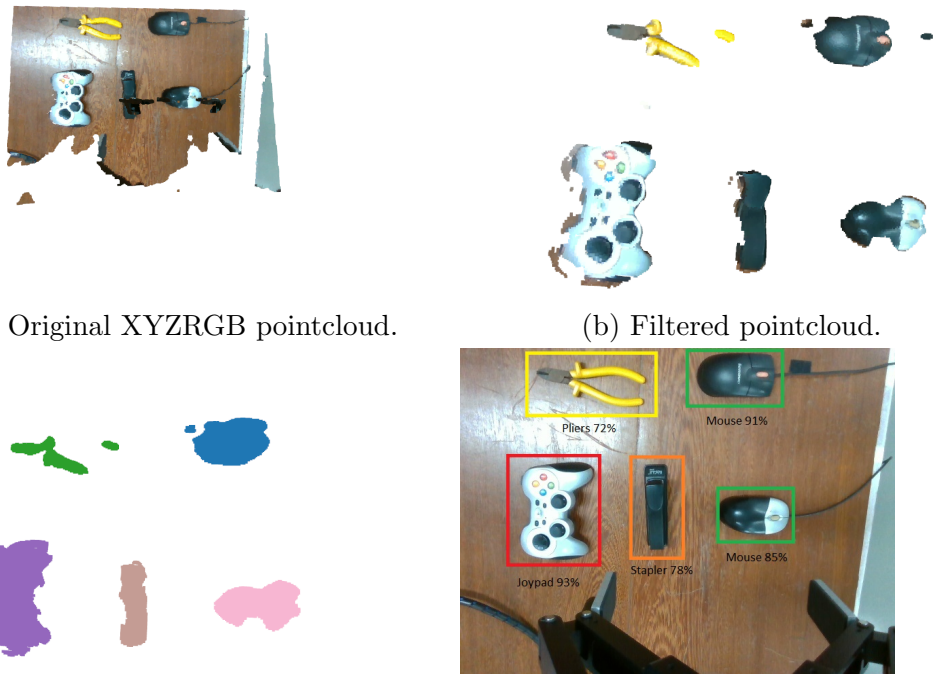


(d) Objects of the class mug.



(e) Objects of the class stapler.

Figure 4.5: Classes and objects of the LARS Classification Dataset.



(a) Original XYZRGB pointcloud. (b) Filtered pointcloud.

(c) Clustered point cloud. (d) Object classification.

Figure 4.6: Classification process for the PEC.

helps the PEC network handle occlusion and defects when classifying objects.

Figure 4.6 provides an example of the classification process, where the network takes a segmented object as input. Object segmentation is achieved by applying Plane Segmentation with RANSAC (RUSU; COUSINS, 2011) to detect and remove the table, resulting in segmented objects within the point cloud. To distinguish between individual objects, the Euclidean Clustering method (RUSU; COUSINS, 2011) is employed. Some defects in the point cloud may be noticeable, such as missing spots, which can occur during the segmentation process. When multiple instances of the same object are present, their confidence rates may vary depending on their similarity to the objects encountered during training. For example, when classifying mice, different models may yield confidence rates ranging from 70% to 95%.

4.4 RESULTS

The validation of PEC is conducted on three datasets: ModelNet10, a widely used public dataset for classification tasks; YCB dataset, a dataset created using the proposed method with objects from the YCB dataset; and LARS Classification Dataset, a dataset comprising objects similar to those found in the laboratory. The input point cloud size for all experiments is set to 64 points, except for PEC_{128} and PEC_{256} which utilize input sizes of 128 points and 256 points, respectively. To examine the impact of different point

groupings, groupings of 4, 8, 16, 32, and 64 points were formatted. Since the ModelNet10 dataset is a dataset composed of 3D meshes of objects instead of point clouds, Poisson-Disk Sampling (YUKSEL, 2015) was used to convert the mesh to a point cloud and subsample the objects.

4.4.1 ModelNet10

The accuracy of the network on the ModelNet10 is presented in Table 4.1, where PEC is compared to other works on the literature, including the state of the art. It can be observed that the group size influences the network's accuracy, and having more points in the input does not necessarily lead to better results. PEC was trained on this dataset with a learning rate of 0.001 for 150 epochs since those were the hyperparameters that returned the best results for this dataset on the experiments done.

Network	Accuracy
PEC_{group4}	78.8%
PEC_{group8}	88.74%
$PEC_{group16}$	89.5%
$PEC_{group32}$	69.8%
$PEC_{nogroup}$	79.4%
$PEC_{128_{group32}}$	92.24%
$PEC_{256_{group64}}$	82.9%
3DShapeNets (WU et al., 2015)	83.5%
OrthographicNet (KASAEI, 2019)	88.56%
VSL (LIU; GILES; ORORBIA, 2018)	91.0%
LonchaNet (GOMEZ-DONOSO et al., 2017)	94.37%
SPNet (YAVARTANOO; KIM; LEE, 2019)	97.25%
RotationN (KANEZAKI; MATSUSHITA; NISHIDA, 2018)	98.46%
VRN Ensemble (BROCK et al., 2016)	97.14%
Panorama-ENN (SFIKAS; PRATIKAKIS; THEOHARIS, 2018)	96.85%
SO-Net (LI; CHEN; LEE, 2018)	95.7%
CurveNet (XIANG et al., 2021)	96.3%
Voxelized Point Clouds (GEZAWA et al., 2022)	93.4%
Shape Self-Correction (CHEN et al., 2021)	95.5%

Table 4.1: Table comparing classification networks on ModelNet10.

4.4.2 YCB Dataset

The accuracy of the network on the YCB dataset is presented in Table 4.2, where the best result was achieved with an input size of 64 points and a group size of 16 points. Interestingly, for this dataset, the input size had minimal influence on the results, as an input size of 256 yielded similar results to an input size of 64. On the other hand, the number of points per group had a more significant impact, with the biggest difference

observed when the points were not organized into groups. The dataset was split into 75% for training and 25% for validation, utilizing a learning rate of 0.01 and trained for 150 epochs. Those values were chosen due to yielding better results from multiple experiments.

Network	Accuracy
PEC_{group4}	93%
PEC_{group8}	97.3%
$PEC_{group16}$	99.3%
$PEC_{group32}$	97.2%
$PEC_{nogroup}$	83.9%
$PEC_{256_{group64}}$	99%

Table 4.2: PEC accuracy on YCB Dataset.

4.4.3 Lars Classification Dataset

The accuracy of the network on the LARS Classification Dataset is presented in Table 4.3, where the best result was achieved with an input size of 64 points and a group size of 16 points. Similar to the YCB dataset, the LARS dataset was split into 75% for training and 25% for validation, using a learning rate of 0.01, and trained for 150 epochs, where similarly to the last experiment, those values yielded the best results for this dataset when compared to other hyperparameters that were tested. Figure 4.7 illustrates the validation accuracy and loss during the training stage. It can be observed that the validation loss starts high due to batch normalization but eventually normalizes, resulting in a faster and more stable training process. Although the graph may appear noisy, it was found to be an effective learning rate that consistently yielded good results. In contrast, Figure 4.8 demonstrates the training process with a suboptimal learning rate. Figure 4.8a depicts the training process when the learning rate is too high, leading to instability and compromising optimal training performance. Conversely, Figure 4.8b shows the opposite scenario, where the learning rate is too low, causing the optimizer to get stuck in local minima for an extended period and requiring more time to reach optimal results.

Network	Accuracy
PEC_{group8}	89.2%
$PEC_{group16}$	92.5%
$PEC_{group32}$	89.1%
$PEC_{nogroup}$	67%
$PEC_{128_{group32}}$	91%
$PEC_{256_{group64}}$	91.5%

Table 4.3: PEC accuracy on LARS Classification Dataset.

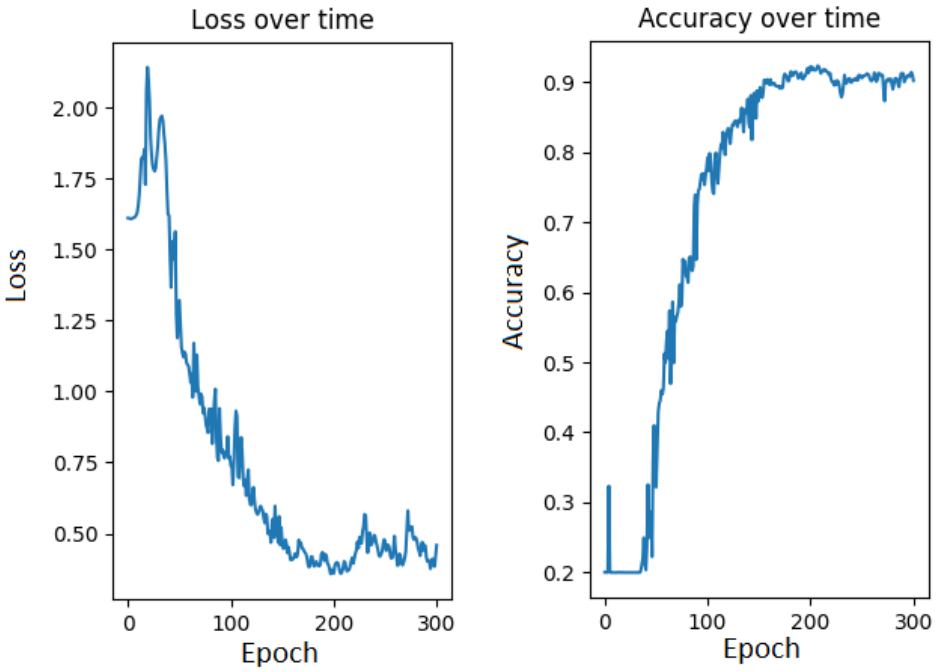
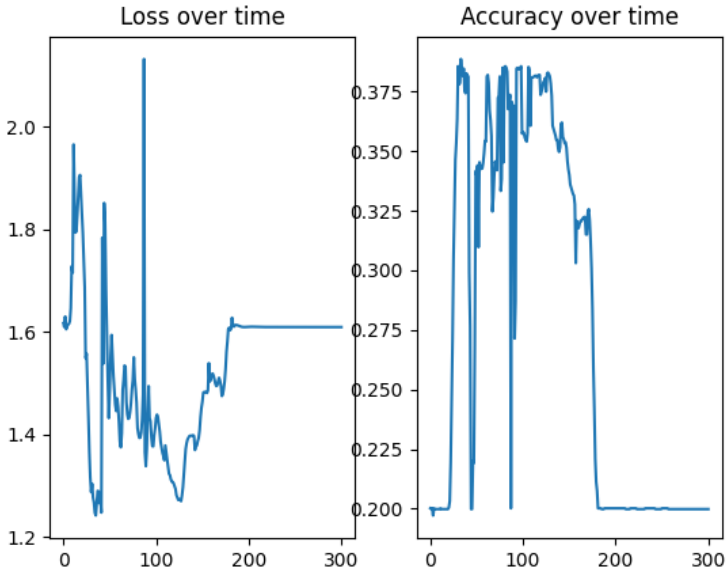


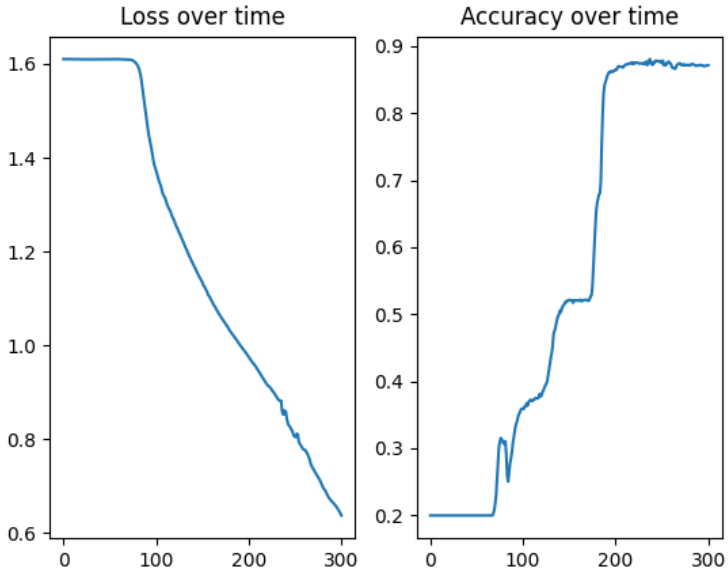
Figure 4.7: Training process of PEC on *LARS classification Dataset*.

4.4.4 Execution and Training Time

Using an RTX 3060ti graphics card, the PEC network with the LARS dataset can perform predictions in approximately 0.002 seconds, while the training step takes 152 seconds when using an input size of 64 points. On the other hand, with a GT850M graphics card, the prediction time is around 0.009 seconds, and the training process takes approximately 250 seconds. The efficiency in training time is attributed to the dataset's characteristics, where each sample contains a low number of points, allowing the entire dataset to fit into memory, even on a laptop GPU like the GT850M. This aspect reduces the need for frequent disk reading during training, resulting in faster training times. It is essential to emphasize that the network's training step is only performed once for a set of object classes. Once trained, the PEC network demonstrates remarkable performance in execution time and can be deployed on low-cost hardware effectively. Additionally, the network's versatility shines through, as it can be retrained as needed to accommodate changes in the desired objects to be classified. This adaptability adds to its practicality and makes it an excellent choice for scenarios where hardware constraints or variations in the target objects may arise.



(a) Training process of PEC on *LARS classification Dataset* when the learning rate is too high.



(b) Training process of PEC on *LARS classification Dataset* when the learning rate is too low.

Figure 4.8: Comparassion of the PEC training process when the learning rate does not have its optimal value.

CHAPTER 5

GRASPING BASED ON LATERAL CURVATURES AND GEOMETRIC PRIMITIVES

This chapter proposes a novel algorithm for 6D pose grasping using lateral curvatures and geometric primitives of point clouds. The algorithm builds upon the works of Zapata et. al. and Jain et. al. (ZAPATA-IMPATA et al., 2019; Jain; Argall, 2016) and offers two significant improvements:

1. The algorithm breaks the object into smaller and more accessible regions to perform the grasp. This approach allows analyzing the entire object in less complex regions, improving upon the algorithm proposed in Zapata et. al. (ZAPATA-IMPATA et al., 2019), which focuses its analysis near the object's center of mass. Moreover, by identifying regions that do not fit the gripper, unnecessary computation time is saved during the analysis.

2. The algorithm enhances the geometric primitive estimation proposed by Jain et. al. (Jain; Argall, 2016), which tries to reduce complex objects to a single primitive. Instead, by breaking the object into multiple regions, the algorithm can assess whether the object can be effectively reduced to a geometric primitive and grasp it accordingly if possible.

The input to the grasping algorithm is a point cloud $p \subset \mathbb{R}^3$ containing only the desired object to be grasped. The algorithm considers a two-finger gripper with gripper stroke W_g and finger width H_g, as illustrated in Figure 5.1. The output of the algorithm is a 6D pose $O \subset \mathbb{R}^6$ that indicates how to grasp the object, with its position aimed at the center of the selected region.

The proposed grasping algorithm operates as follows:

- It employs PCA to establish the object's orientation in relation to the camera, as suggested in Zapata et. al. and Jain et. al. (ZAPATA-IMPATA et al., 2019; Jain; Argall, 2016).

- The object is then aligned with the origin axis based on its orientation.

- Grasping regions are generated based on the H_g value, which facilitates further analysis.

- The algorithm computes the curvature of each region and checks if they share the same geometry, using Jain et. al.'s work (Jain; Argall, 2016) as a foundation for estimating each geometry.

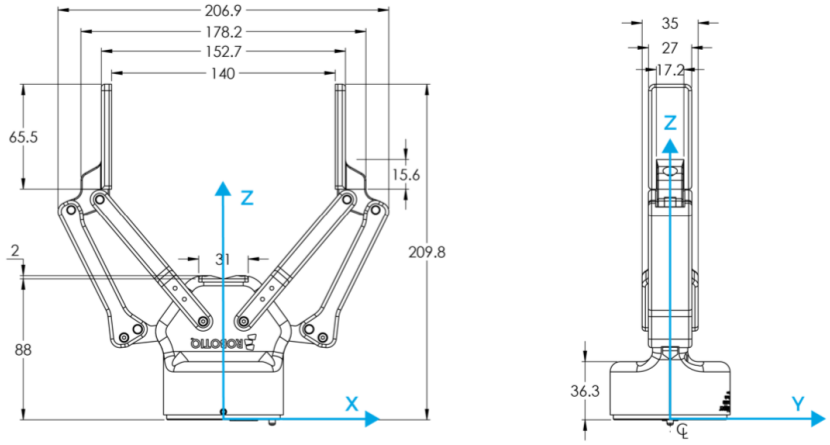


Figure 5.1: Robotiq 2F-140 general dimensions: gripper stroke $W_g = 140$ mm and finger width $H_g = 27$ mm.

- If the object can be reduced to a primitive by verifying if each region has the same geometric primitive and width, the algorithm grasps the object by its centroid.
- Otherwise, it calculates the lateral curvature of each region.
- The algorithm then returns the pose of the region with the smallest curvature that fits inside the gripper, enabling successful grasping.

5.1 REGION GENERATION

The number of regions K is defined using the object height H_o and the gripper dimensions:

$$K = \frac{H_o}{H_g} \tag{5.1}$$

where H_o can be obtained using the following equation, where y_{max} and y_{min} are the topmost and bottommost points, respectively:

$$H_o = y_{max} - y_{min} \tag{5.2}$$

To generate the grasping K regions, the following algorithm is used, where $fp(p, y_{min}, y_{max})$ is PassThrough filter used to reduce the field of view of the point cloud p between (y_{min}, y_{max}) on the y-axis and p_i is the generated region, where $i = 1, ..., K$:

Algorithm 1 Algorithm to generate the possible grasping regions

$y \leftarrow y_{min}$
$y_f \leftarrow y_{min} + h_g$
while $i < K$ **do**
  $p_i \leftarrow fp(p, y, y_f)$
  $y \leftarrow y_f$
  $y_f \leftarrow y_f + h_g$
  $i \leftarrow i + 1$
end while

If $H_o \leq H_g$, the object is considered too small, and grasping will be performed at its center. Otherwise, the object is broken into smaller regions. Since some objects have regions that can be separated into multiple regions, for each region found, a clustering algorithm is applied to separate them when applicable, where Euclidean Cluster Extraction (RUSU; COUSINS, 2011) was used. This way, a complex object can be broken into smaller and more accessible regions, as shown in Figure 5.2 with the joypad example.

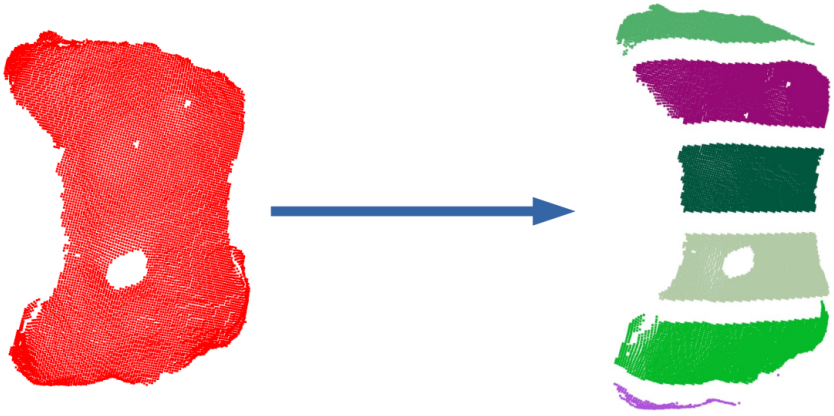


Figure 5.2: A joypad broken into regions.

5.2 CURVATURE CALCULATION

The point cloud surface curvature is obtained using the method proposed by Jain et. al. and Zapata et. al. (Jain; Argall, 2016; ZAPATA-IMPATA et al., 2019). First, the centroid of each region is calculated using the equation below, where N_i is the number of points in the region, with $i = 1, \ldots, N_i$, p_{oi} is a point of the point cloud p_i, and p_c is the centroid of the region:

$$p_c = \frac{1}{N_i} \sum_{i=1}^{N_i} p_{oi}, \quad p_{oi} \in p_i. \tag{5.3}$$

Then, the covariance matrix is obtained:

$$C = \frac{1}{N_i} \sum_{i=1}^{N_i} (p_{oi} - p_c) \cdot (p_{oi} - p_c)^T, \tag{5.4}$$

and the eigenvalues λ and eigenvectors for three dimensions $\vec{v}$ are obtained by solving the equation:

$$C \cdot \vec{v}_t = \lambda_t \cdot \vec{v}_t, \quad t \in 1, 2, 3, \tag{5.5}$$

and by using PCA, the curvature σ_{j_*} is defined as:

$$\sigma_{j_*} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}. \tag{5.6}$$

5.3 GEOMETRIC PRIMITIVE REDUCTION

The geometric primitives, proposed by Jain et. al. (Jain; Argall, 2016), are estimated through the curvature around each point of the region using the Equations 5.6 and 5.7:

$$\Delta = \frac{M^*}{M}. \tag{5.7}$$

where M^* is the number of points $j_* \in j_i$ where $\sigma_{j_*} \leq 0.01$ and M is the total number of points from the point cloud. This way, the region geometric primitive can be defined as seen below:

Sphere	$0.0 \leq \Delta \leq 0.10$
Cylinder	$0.10 < \Delta \leq 0.40$
Box	$0.40 < \Delta \leq 1.0$

Table 5.1: Geometric primitive classification

5.4 REGION SELECTION

The best region to grasp is defined considering whether each region has the same geometric primitive and a similar length, within a margin of error due to noises and measurement errors in the point cloud. If they have this, consider the entire object as a geometric primitive and grasp it at its centroid. The region length L_i is defined by:

$$L_i = x_{\max p_i} - x_{\min p_i}, \tag{5.8}$$

where $x_{\max p_i}$ and $x_{\min p_i}$ are the maximum and minimum values of each generated region p_i on the x-axis. The x-axis represents the horizontal axis, p_i represents the region, with $i = 0, 1, \ldots, K$, and K is the number of regions.

If this condition fails, the curvature around $x_{\max p_i}$ and $x_{\min p_i}$ is calculated, and a score called P_i is defined as:

$$P_i = \sigma_{xmaxpi} + \sigma_{minpi}, \tag{5.9}$$

where σ_{xmaxpi} is the curvature around $x_{\max p_i}$, the point on the right extremity of the object, and σ_{minpi} is the curvature around $x_{\min p_i}$, the point on the left extremity of the object.

The best region to grasp is the one where P_i is minimal, and the condition $L_i \leq W_g$ is valid to avoid regions that do not fit inside the gripper. In the last step, PCA is applied to the region as it may have a different orientation than the object, as seen in Figure 5.3.

The output of the algorithm will be the 6D pose of the region to be grasped, denoted as p_r.

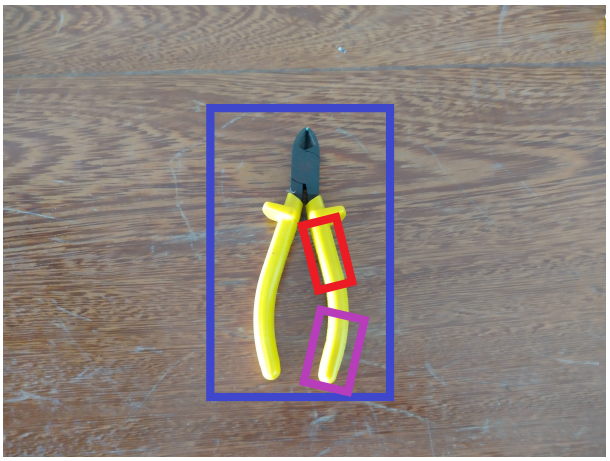


Figure 5.3: An object with regions that have a different orientation than the complete object orientation.

5.5 RESULTS

This section presents the results obtained using the proposed grasping algorithm. Section 5.5.1 shows the validation of the grasping algorithm in the simulated environment, where the system operates as shown in Section 3.2. The algorithm used to estimate the 6D pose and segment the object was Segmentation-driven 6D object pose estimation (HU et al., 2019). For the initial experiments in the Gazebo simulator, the gripper was not used due to collision physics problems that made grasping tasks unviable when using the default physics configuration. Webots was also used for validation since it has better collision physics without the need for plugins, allowing simulated grasping to occur.

Section 5.5.2 focuses on the experimental validation of the grasping algorithm using a system similar to the one described in Section 3.2, but without the classification part. The main objective was to evaluate the performance of the grasping algorithm on five distinct objects, each presenting different geometries and challenges. Additionally, an experiment was conducted to assess the potential benefits of using an active visual sensor, where the robotic manipulator is moved around the object to create a complete 3D point cloud without encountering occlusion issues. However, the results of this experiment did not show significant improvements. The experimental setup involved using the UR5 robotic manipulator equipped with a Robotiq 2F-140 gripper and an RGB-D visual sensor, specifically the Intel Realsense D435.

In Section 5.5.3, the full system proposed in Section 3.2 is utilized for the task of selective grasping. The hardware used is the same as seen in the validation of the grasping algorithm.

Section 5.5.4 provides the execution time of the grasping algorithm and an analysis of its time complexity.

A video demonstrating further experiments can be viewed in the link ⟨https://youtu.be/h6z5AXT9CZE⟩.

5.5.1 Simulated Results

For the initial validation of the proposed grasping system, the simulator Gazebo was utilized to test its viability. In the simulated environment, the robotic manipulator UR5 was employed, as depicted in Figure 5.4. The simulated environment also includes an RGB-D sensor, represented by the cube above the robotic arm, with parameters similar to those found in the Kinect documentation. The initial pose of the robotic arm and the RGB-D sensor in the simulated environment is defined as x = *-0.4m,* y=*-0.01m,* z=*0.5m,* φ=*3.14 rad,* θ=*0 rad e* ψ=*1.57 rad*, where x, y, z represent the position coordinates of the end effector of the robotic manipulator in the world frame, φ represents the rotation on x *(roll)* axis, θ represents the rotation on the y *(pitch)* axis and ψ represents the rotation on the z *(yaw)* axis.

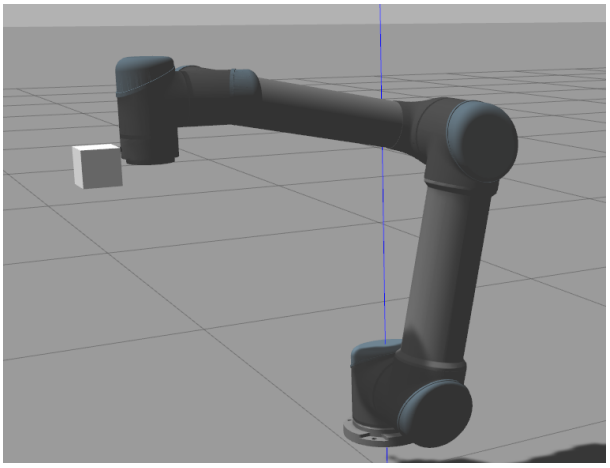


Figure 5.4: UR5 on the Gazebo simulator.

For the simulated experiments, the objects from the YCB dataset were utilized, as it provides the 3D model of each object, making it suitable for testing the proposed grasping system. The neural network used for 6D pose estimation is trained with this dataset, making it compatible with the objects used in the simulation. The Segmentation-driven 6D Object Pose Estimation (HU et al., 2019) algorithm was selected for 6D pose estimation due to its lower rate of false positives and more accurate pose estimation, outperforming other tested networks in the simulated environment.

Figures 5.5, 5.6, 5.7 and Figure 5.8 depict the regions selected for grasping four different objects. In these figures, the red region indicates the optimal grasping region as

determined by the grasping algorithm, except for objects that have a primitive geometry, where we just grasp by its center.

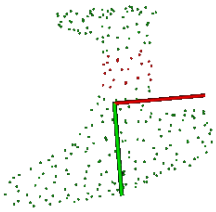
For the drill case, the grasping region is located on its handle, similar to how a human would grasp it. The algorithm identifies this handle as the most appropriate region for grasping due to its linear shape and ergonomic design.

In the case of the bleach container, the grasping region near the bottom is chosen because of its more linear and graspable structure on its laterals. The algorithm recognizes this region as an ideal location to achieve a secure grip.

The clamp is an object with handles that can be separated into two distinct grasping regions due to the clustering algorithm applied on each region generated. Due to this, the algorithm selects a single handle to grasp since it will detect the single handle as the best place to grasp.

The tomato soup can is grasped by its centroid. The algorithm identifies the object's geometry as resembling a cylinder, and therefore, it selects the center for the grasp.

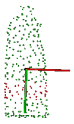
(a) Original drill 3D mesh



(b) Best region to grasp the drill.

Figure 5.5: 3D mesh model and the best region to grasp for the drill

(a) Original bleach 3D mesh



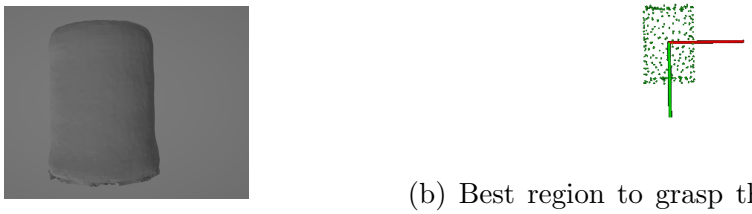
(b) Best region to grasp the bleach.

Figure 5.6: 3D mesh model and the best region to grasp for the bleach

In the first grasping experiment, the target object was the tomato soup can, positioned at x = *-0.48m, y=-0.07m, z=0.40m,* φ*=3.14 rad,* θ*=0 rad e* ψ*=1.57 rad.* The grasping process is illustrated in Figure 5.9, and the final grasp position achieved was x = *-0.48m, y=-0.08m, z=0.43m,* φ*=3.10 rad,* θ*=0.04 rad e* ψ*=1.63 rad.* The resulting final error in the grasp position was x = *1.47%, y=16%, z=7.5%,* φ*=1.2%,* θ*=4% rad e* ψ*=3.8%.*

(a) Original clamp 3D mesh. (b) Best region to grasp the clamp.

Figure 5.7: 3D mesh model and the best region to grasp for the clamp



(a) Original tomato soup can 3D mesh. (b) Best region to grasp the tomato soup can.

Figure 5.8: 3D mesh model and the best region to grasp for the tomato soup can

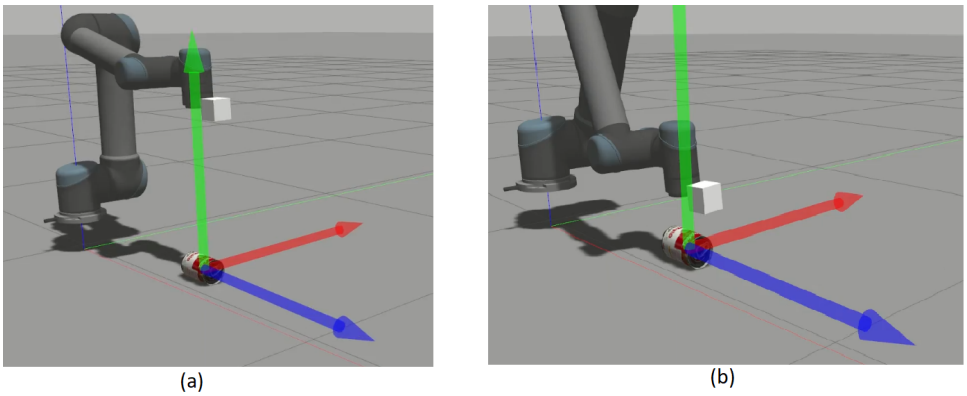


Figure 5.9: End effector pose in the grasp to grasp the tomato soup can, where: (a) is the intermediary pose and (b) is the final pose.

In the next experiment, the task was to grasp the bleach, positioned at x = *-0.50m, y=0.02m, z=0.30m, φ=-2.5 rad, θ=0 rad e ψ=0.7 rad.* The robotic manipulator suc-

cessfully grasped the object at the pose x = *-0.51m, y=0.02m, z=0.34m,* φ*=-2.72 rad,* θ*=0.023 rad e* ψ*=0.714 rad*, resulting in an error of x = *2.2%, y=10%, z=13%,* φ*=8.8%,* θ*=2.3% rad e* ψ*=2% rad.*

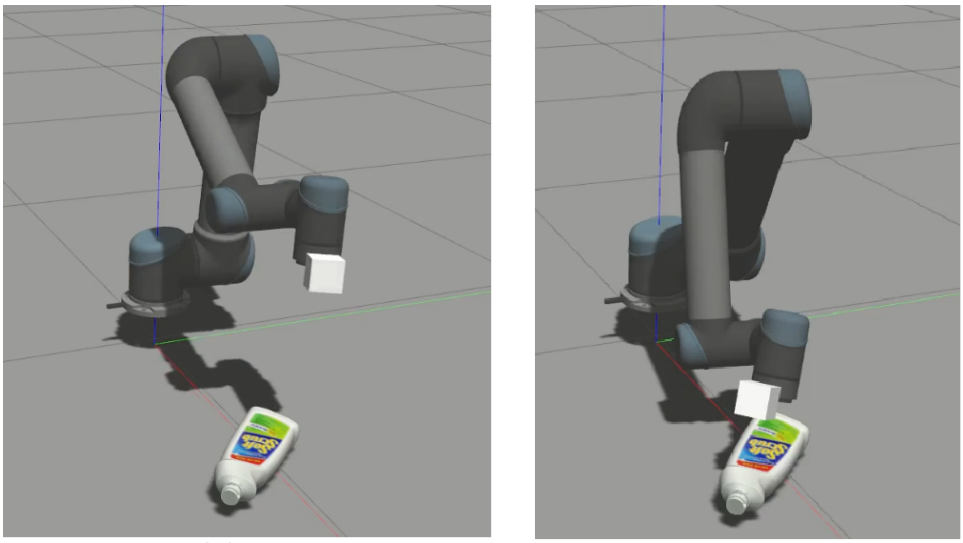


Figure 5.10: End effector pose in the grasp to grasp the bleach, where: (a) is the intermediary pose and (b) is the final pose.

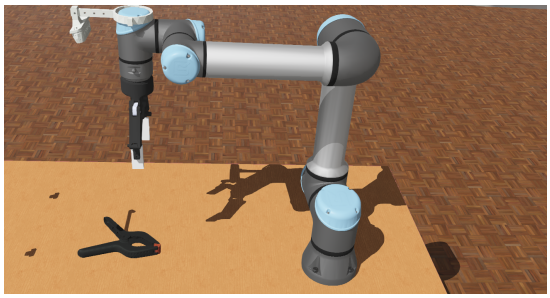
During the Gazebo experiments, it became apparent that the selected neural network exhibits low orientation error but relatively high position error, particularly on the y and z axes. The next experiment utilized Webots, which provide better physics for collision handling, enabling more realistic simulated grasping. Figure 5.11 showcases the four stages involved in grasping a clamp in the simulation.

Despite the presence of pose estimation errors that may affect precision during grasping, resulting in slight deviations from the best-estimated position, the Webots simulation demonstrated successful grasping without any major issues, like missing the object, for the tested objects.

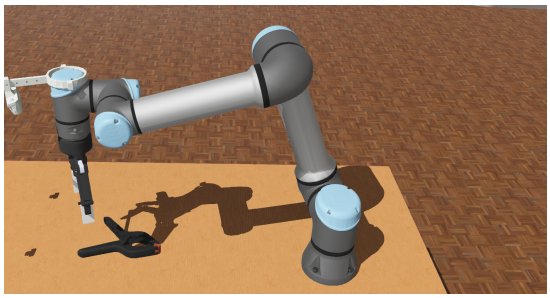
5.5.2 Experimental Validation Results

Figure 5.12 shows step-by-step how the grasp is performed. The point cloud of the objects and their respective grasping regions can be seen in Figure 5.13, showcasing the performance of the grasping stage with objects of varying complexities:

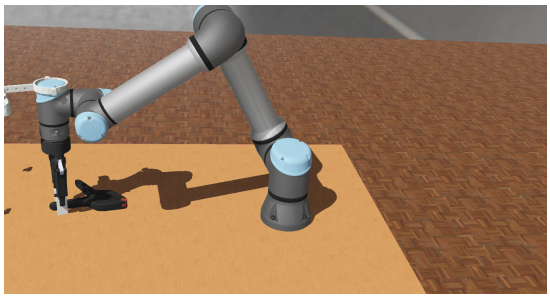
- Joypad: A complex object where the gripper needs to grasp close to its center for better stability.
- Pliers: For the pliers, both the clusters and the original region are considered as possible grasping options since some handles may not be ideal to be grasped individually. The grasping algorithm chose to grasp both handles simultaneously.



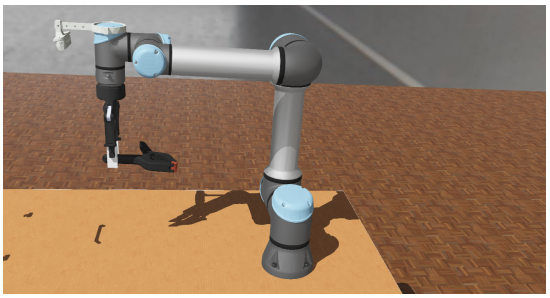
(a) Initial position.



(b) Pre grasping position



(c) Grasping position.



(d) Final position.

Figure 5.11: The four grasping stages of grasping the clamp.

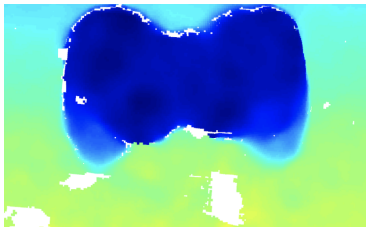
- Plug Adapter: A rectangular object that uses the wall plug to stay upright, causing it to have an orientation about the table. Due to its rectangular geometry, the gripper grasps it by its center.
- Wire Cutter: The wire cutter has predominantly planar geometry and lacks depth. Therefore, the plane segmentation considers only the handles.
- Cylinder: Another object with a simple geometry, which is grasped by its center.

Figure 5.12 shows an example of the selective grasping algorithm acting on the controller object alone. This experiment can be seen on the link ⟨https://youtu.be/OeepuDNFY9Y⟩ with further details.

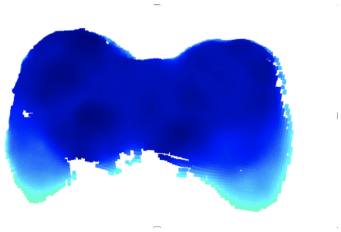
To benchmark the proposed grasping system, a Generic Grasping Algorithm (GGA) is used for comparison. The proposed GGA is a grasp algorithm that will only grasp the object by its center, without any kind of analysis and keeps the gripper perpendicular to the table. For simple objects, such as the cylinder and the plug adapter, the difference in success rate is minimal. The success of the grasp depends more on other factors, such as correct pose estimation and object dimensions. Table 5.2 shows the results of grasping an object twenty times in different poses to demonstrate the efficiency of the algorithm on different poses where a more significant gap between the proposed grasping algorithm and the GGA is noticeable for more complex objects.

A common problem with RGB-D sensors is that the point cloud, when seen from a single angle, is considered to have 2.5D due to its susceptibility to occlusion. To address this limitation, an approach involving moving the robotic manipulator around the object

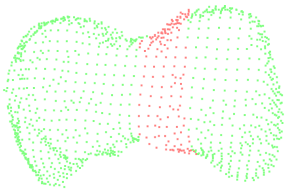
(a) Initial pose of the joypad.



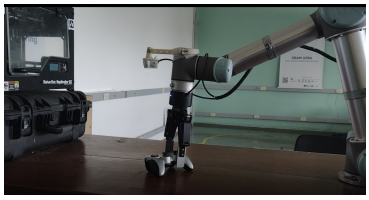
(b) Original point cloud.



(c) Filtered point cloud.



(d) Region for grasping in red.



(e) Executing the grasping.

(f) Final pose.

Figure 5.12: Sequence of stages to perform a grasp.

Object	Proposed	GGA
Joypad	90%	40%
Cylinder	95%	85%
Plug Adapter	100%	100%
Pliers	100%	65%
Wire cutter	95%	70%
Wire cutter (Open)	85%	0%

Table 5.2: Table comparing the proposed grasping system vs. a Generic Grasping Algorithm (GGA).

was explored to create a full 3D point cloud without occlusion issues. Figure 5.14 illustrates the poses used to generate the full 3D point cloud of the object, where the poses were selected through experimentation. However, a challenge arose from the position of the camera in relation to the gripper, resulting in different orientations between the camera and the end effector in some poses. To rectify this misalignment, ICP (BESL; MCKAY, 1992) was employed, since it can be used to align two points cloud of unknown transformation between each other. Initially, the table was removed using Plane Segmentation, similar to the pre-grasping process, as ICP was not converging correctly with the table obstructing the field of view. After table removal, point-to-point ICP was applied with 100 iterations and a maximum error tolerance of 1 centimeter. Subsequently, all the angles were merged into a single object, and a Voxel Grid Downsample (RUSU; COUSINS, 2011) was used to merge duplicate or close points.

Figure 5.15 showcases the comparison between the single-shot point cloud and the point cloud generated from multiple angles. The point cloud generated with the active

visual sensor demonstrates more details and provides data on previously occluded regions, resolving some of the issues faced in single-angle capture. However, a limitation of this approach was identified when dealing with objects with simple geometry, such as cubes or cylinders. ICP encountered difficulties in aligning such objects accurately. Overall, for the tested objects, this approach did not yield significant improvements in results.

5.5.3 Selective Grasping Results

Selective grasping allows the user to choose an object of interest from a given environment with other objects. Figures 5.16 and 5.17 show the system successfully grasping the joypad and the staples, respectively. Those experiments can be seen in the footnotes⟨https://youtu.be/5o0LnLQrWyo⟩ and ⟨https://youtu.be/6CfazJfU7yA⟩, respectively.

It is important to note that during the testing of the selective grasping system, objects similar to those used in the simulation training stage were used. A problem that was observed while doing the experiments was that the quality of the point clouds obtained from the sensors played a significant role in the system's success. The point clouds acquired by the sensors often contain noise and defective regions, which can lower the confidence score of the classification compared to the simulated point clouds. However, despite these challenges, the system demonstrated its capability to handle a diverse set of objects effectively.

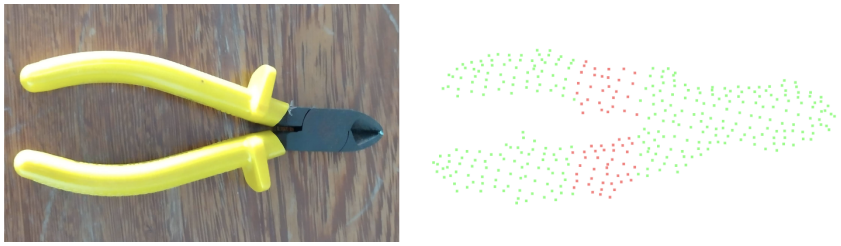
5.5.4 Time Complexity and Execution Time

The grasp algorithm exhibits efficient performance, taking on average only 0.0018s to generate a grasp for the objects used in these experiments, running on a CPU Ryzen 5 3600.

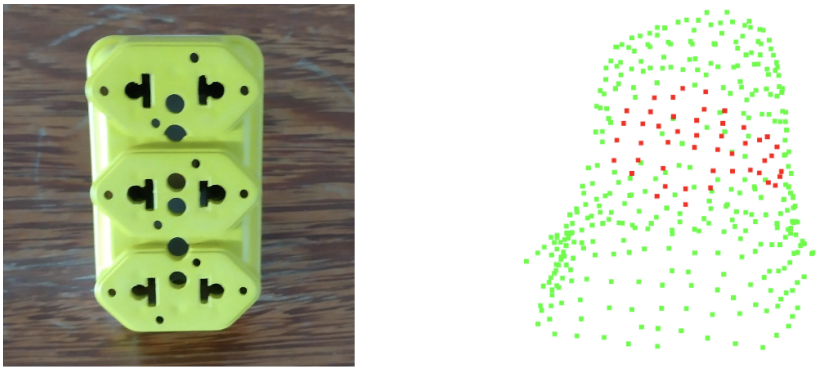
Regarding time complexity, the main challenge lies in the creation and search of a group of points using a K-D Tree (BENTLEY, 1975). The point cloud data is not inherently ordered like an RGB image, necessitating the use of a k-d tree for neighbor point search to estimate the curvature of a region. The k-d tree is a nonlinear algorithm with a complexity of $O(n \log n)$ for tree creation and search.

However, breaking the object into smaller regions proves to be advantageous, as the k-d tree creation and search are performed on smaller objects. As the k-d tree's execution time grows nonlinearly with the increase in data size, using smaller regions offers a significant advantage over dealing with the entire object when it comes to execution time.

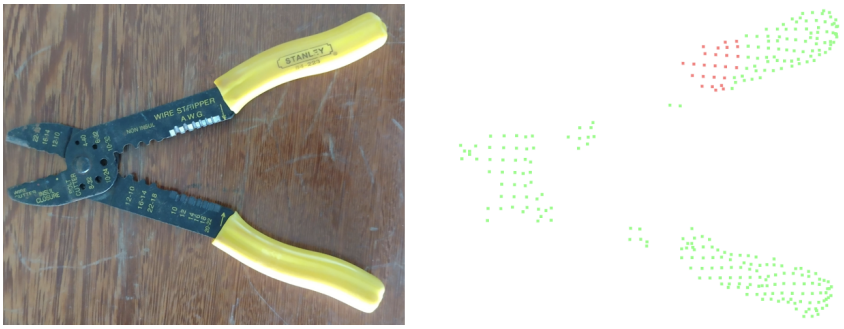
(a) Joypad.



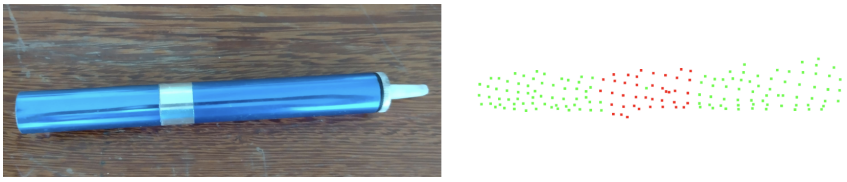
(b) Pliers.



(c) Plug adapter.



(d) Wire cutter.



(e) Cylinder.

Figure 5.13: RGB images and their respective point clouds. Objects with simple geometry are grasped by their centroids.

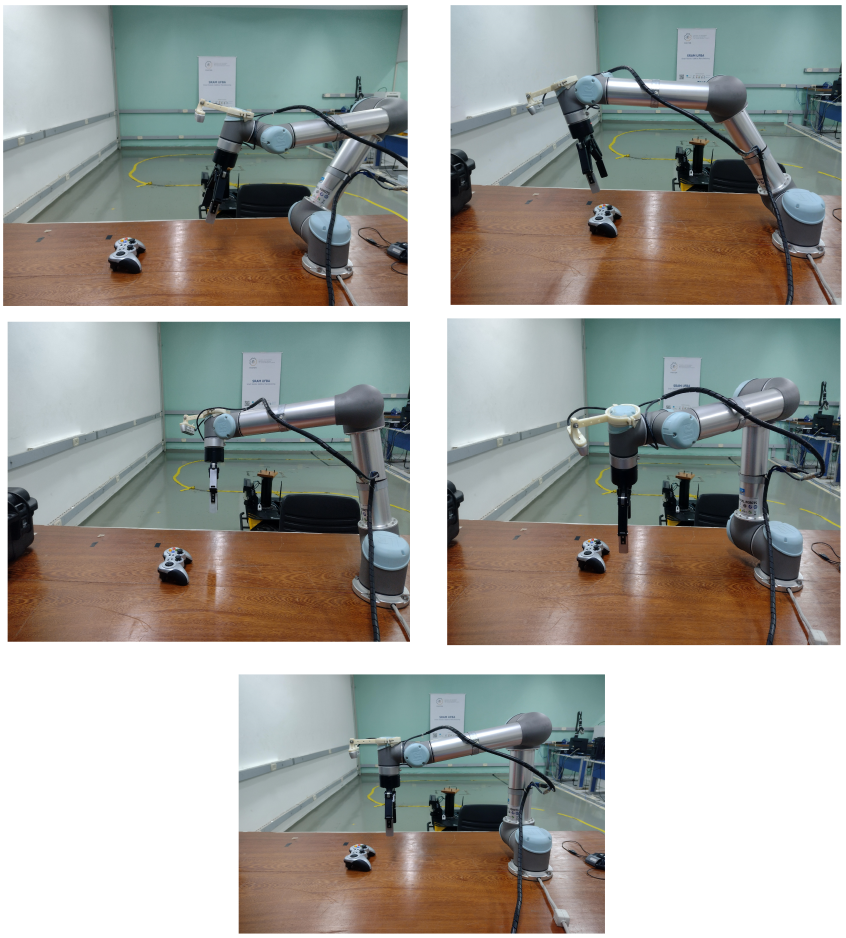
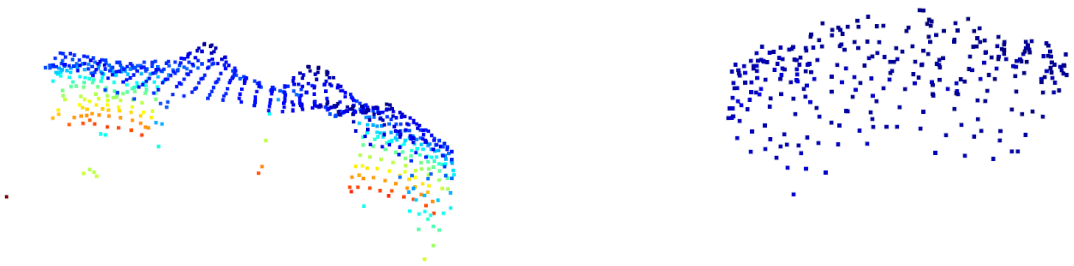


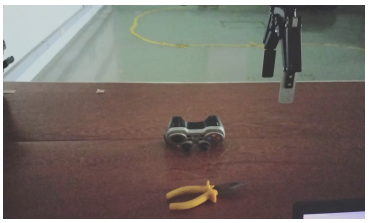
Figure 5.14: The poses used to generate the full 3D pose of the object.



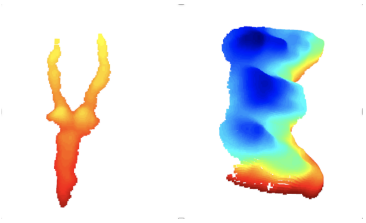
(a) Point Cloud from a single shot point cloud.

(b) Point cloud generated from multiple angles.

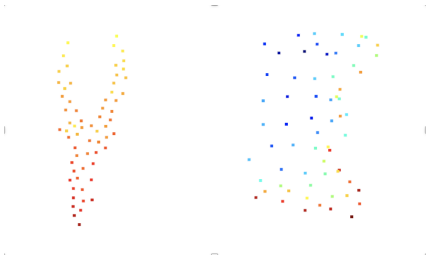
Figure 5.15: Comparison between a single shot point cloud and a point cloud merged from multiple angles, where the point cloud that was generated from multiple angles has a more detailed and complete point cloud.



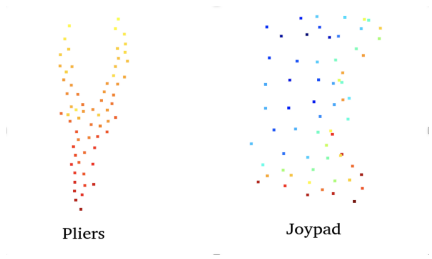
(a) Initial pose.



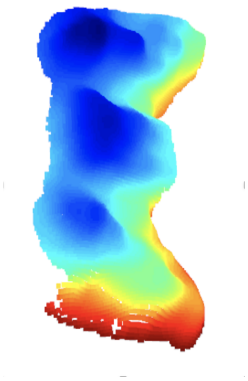
(b) Filtered point clouds.



(c) Pre-processed objects.



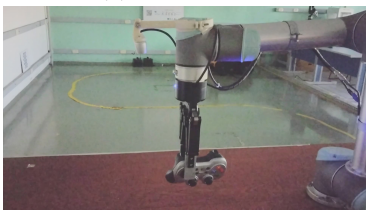
(d) Classified objects.



(e) Selected object.

(f) Grasp region.

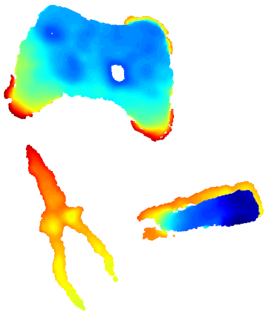
(g) Grasping.



(h) Final pose.

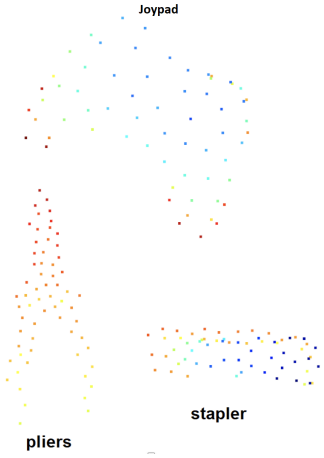
Figure 5.16: Selective Grasping. The environment has two objects (pliers and a joypad), and the objective is to grasp the joypad.

(a) Initial pose.

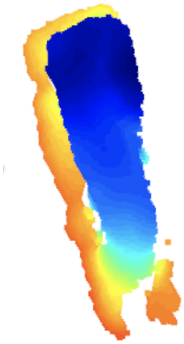


(b) Filtered point clouds.

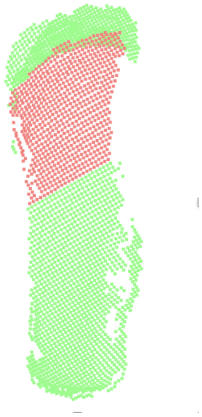
(c) Pre-processed objects.



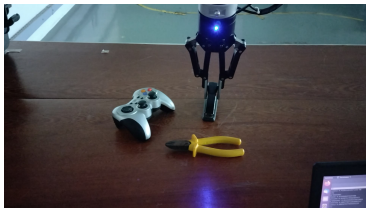
(d) Classified objects.



(e) Selected object.



(f) Grasp region.



(g) Grasping.

(h) Final pose.

Figure 5.17: Selective Grasping. The environment has three objects (pliers, staples, and a joypad), and the objective is to grasp the staples. This experiment can be seen in the footnote

CHAPTER 6

CONCLUSION

This research endeavor introduces a selective grasp system optimized for deployment on cost-effective hardware, capable of executing in approximately 0.004 seconds, 0.002 seconds for classification and 0.0018 seconds for pose generation of feasible grasp configuration. The presented grasping algorithm, capable of identifying unfamiliar objects, has augmented and overcome the limitations of existing algorithms, thereby enhancing their utility. Additionally, a specialized deep learning neural network, known as Point Encoder Convolution (PEC), was devised to facilitate object classification, empowering the grasping algorithm to make informed selections for optimal grasp candidates. Two distinct systems were developed to validate the proposed concepts, encompassing both simulation and real-world experimental scenarios.

The PEC neural network is characterized by its simplicity and efficiency, boasting a modest parameter count and expedited training and inference times. This attribute enables training and classification tasks even on outdated or budget-constrained hardware, without incurring prolonged processing durations. To further expedite the dataset creation process, an innovative autonomous dataset generation method was introduced. This approach obviates the need for manual labeling, expediting dataset creation, and minimizing labor requirements. Furthermore, a data preprocessing algorithm was integrated into the framework to refine the training and generalization processes. The training on various datasets, including two datasets generated using the proposed method to generate datasets and a publicly available dataset, demonstrated substantial classification accuracy. Achieving 92.24% accuracy on the public dataset with ten classes and 99.3% and 92.5% on the method-generated datasets underscore the effectiveness of the proposed methodology. While PEC exhibits limitations in scaling to complex classification tasks involving numerous classes, it excels in rapid processing for a limited class set, particularly beneficial for high repeatability robotic tasks.

The grasp algorithm's fundamental principle rests on dissecting intricate objects into more manageable regions. Upon this foundational breakdown, the algorithm investigates the object's potential for reduction into geometric primitives. If feasible, the object is grasped through its centroid. Alternatively, when such simplification proves impractical, the algorithm systematically generates and evaluates potential grasp regions, selecting the optimal candidate for execution. This approach was thoroughly evaluated across diverse scenarios, involving two different grasping systems in distinct environments. The algorithm's robust performance across varying object detection and pose estimation algorithms showcases its adaptability and independence from the specific techniques employed for these tasks. Experimental validation demonstrated the algorithm's efficacy, achieving an average success ratio of 94% across all tested objects. Even upon excluding objects with simpler geometries, the algorithm's average success ratio remained a commend-

able 91.25%. An investigation into the utilization of an active visual sensor, capturing multi-angle images for 3D object representation, yielded inconclusive results as it did not significantly enhance grasp success ratios.

Furthermore, to consolidate the combined algorithms, experiments were conducted where the robotic manipulator selected objects within the environment and successfully grasped them.

In conclusion, this research has accomplished its objective of cultivating a selective grasp system tailored for deployment on cost-effective hardware. By introducing a grasping algorithm and leveraging deep learning techniques, this work has advanced the capabilities of grasping algorithms and object classification methodologies. The proposed PEC neural network and grasping algorithm have been rigorously evaluated through simulations and real-world experiments.

For future works, the current algorithm would greatly benefit from a CUDA implementation. Since we already load the data to use on the PEC, this data already loaded could be utilized on the grasping algorithm as well to speed up costly operations, like the kd-tree. The proposed neural network could also be expanded to a multiclass classification algorithm and tested if its architecture could be used for semantic segmentation. The grasp algorithm could be validated on tasks where two robotic manipulators work together to move big or heavy objects since the algorithm allows the selection of different regions for each manipulator to grasp and avoid collisions between them.

BIBLIOGRAPHY

ARRAIS, R. et al. Application of the open scalable production system to machine tending of additive manufacturing operations by a mobile manipulator. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 11805 LNAI, p. 345 – 356, 2019. Cited by: 12.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 18, n. 9, p. 509–517, sep 1975. ISSN 0001-0782.

BESL, P.; MCKAY, N. D. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 14, n. 2, p. 239–256, 1992.

BRIOT S.; KHALIL, W. *Dynamics of Parallel Robots: From Rigid Bodies to Flexible Elements*. 1ª. ed. : Springer International Publishing, 2015.

BROCK, A. et al. Generative and discriminative voxel modeling with convolutional neural networks. *ArXiv*, abs/1608.04236, 2016.

Carvalho de Souza, J. P. et al. Reconfigurable grasp planning pipeline with grasp synthesis and selection applied to picking operations in aerospace factories. *Robotics and Computer-Integrated Manufacturing*, v. 67, p. 102032, 2021. ISSN 0736-5845.

CHEN, Y. et al. Shape self-correction for unsupervised point cloud understanding. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021. p. 8382–8391.

Ferreira Neto, N. A. et al. Low-latency perception in off-road dynamical low visibility environments. *Expert Systems with Applications*, v. 201, p. 117010, 2022. ISSN 0957-4174.

GAO, G. et al. 6d object pose regression via supervised learning on point clouds. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020. p. 3643–3649.

GEZAWA, A. et al. A voxelized point clouds representation for object classification and segmentation on 3d data. *The Journal of Supercomputing*, v. 78, 01 2022.

GOMEZ-DONOSO, F. et al. Lonchanet: A sliced-based cnn architecture for real-time 3d object recognition. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017. p. 412–418.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. : MIT Press, 2016. ⟨http://www.deeplearningbook.org⟩.

GUALTIERI, M. et al. High precision grasp pose detection in dense clutter. In: IEEE. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016. p. 598–605.

HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020.

HU, Y. et al. Segmentation-driven 6d object pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019. p. 3385–3394.

Jain, S.; Argall, B. Grasp detection for assistive robotic manipulation. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016. p. 2015–2021.

KANEZAKI, A.; MATSUSHITA, Y.; NISHIDA, Y. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018. p. 5010–5019.

KASAEI, S. H. M. Orthographicnet: A deep learning approach for 3d object recognition in open-ended domains. *ArXiv*, abs/1902.03057, 2019.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, v. 81, 02 2009.

LI, J.; CHEN, B. M.; LEE, G. H. So-net: Self-organizing network for point cloud analysis. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 9397–9406, 2018.

LI, Y. et al. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, v. 31, p. 820–830, 2018.

LIU, M. et al. Frame mining: a free lunch for learning robotic manipulation from 3d point clouds. In: *6th Annual Conference on Robot Learning*. 2022.

LIU, S.; GILES, L.; ORORBIA, A. Learning a hierarchical latent-variable model of 3d shapes. In: *2018 International Conference on 3D Vision (3DV)*. 2018. p. 542–551.

LIU, W. et al. Ssd: Single shot multibox detector. In: LEIBE, B. et al. (Ed.). *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016. p. 21–37. ISBN 978-3-319-46448-0.

MAHLER, J. et al. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, p. 1–8, 2018.

MOENNING, C.; DODGSON, N. A. Fast marching farthest point sampling. In: *Eurographics*. 2003.

MONTEIRO, F. et al. Simulating real robots in virtual environments using nvidia's isaac sdk. In: *Anais Estendidos do XXI Simpósio de Realidade Virtual e Aumentada*. Porto Alegre, RS, Brasil: SBC, 2019. p. 47–48. ISSN 0000-0000.

MORAN, M. Evolution of robotic arms. *Journal of Robotic Surgery*, v. 1, p. 103–111, 06 2007.

MOUSAVIAN, A.; EPPNER, C.; FOX, D. 6-dof graspnet: Variational grasp generation for object manipulation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, p. 2901–2910, 2019.

OLIVEIRA, D. M. de; CONCEICAO, A. G. S. Preensão de objetos em 6d utilizando redes neurais convolucionais e curvaturas. *Procedings do XV Simpósio Brasileiro de Automação Inteligente*, 2021.

OLIVEIRA, D. M. de; CONCEICAO, A. G. S. A fast 6dof visual selective grasping system using point clouds. *Machines*, MDPI AG, v. 11, n. 5, p. 540, May 2023. ISSN 2075-1702.

OLIVEIRA, D. M. de; VITURINO, C. C. B.; CONCEICAO, A. G. S. 6d grasping based on lateral curvatures and geometric primitives. In: *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*. 2021. p. 138–143.

OSA, T.; PETERS, J.; NEUMANN, G. Hierarchical reinforcement learning of multiple grasping strategies with human instructions. *Advanced Robotics*, Taylor Francis, v. 32, n. 18, p. 955–968, 2018.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019. p. 8024–8035. Disponível em: ⟨http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf⟩.

QI, C. R. et al. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. p. 652–660.

QI, C. R. et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: *NIPS*. Red Hook, NY, USA: Curran Associates Inc., 2017. p. 5105–5114. ISBN 9781510860964.

QU, Z. et al. An improved yolov5 method for large objects detection with multi-scale feature cross-layer fusion network. *Image and Vision Computing*, v. 125, p. 104518, 2022. ISSN 0262-8856.

RAO, Y. et al. Semantic point cloud segmentation using fast deep neural network and dcrf. *Sensors*, v. 21, n. 8, 2021. ISSN 1424-8220.

RUSU, R. B.; COUSINS, S. 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: , 2011.

SFIKAS, K.; PRATIKAKIS, I.; THEOHARIS, T. Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval. *Computers & Graphics*, v. 71, p. 208–218, 2018. ISSN 0097-8493.

SIROHI, K. et al. Efficientlps: Efficient lidar panoptic segmentation. *IEEE Transactions on Robotics*, v. 38, p. 1894–1914, 2022.

WANG, C. et al. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020. p. 10059–10066.

WANG, C. et al. Densefusion: 6d object pose estimation by iterative dense fusion. 2019.

WANG, L. et al. Hierarchical policies for cluttered-scene grasping with latent plans. *IEEE Robotics and Automation Letters*, v. 7, n. 2, p. 2883–2890, 2022.

WU, Z. et al. 3d shapenets: A deep representation for volumetric shapes. In: . 2015. p. 1912–1920.

XIANG, T. et al. Walk in the cloud: Learning curves for point clouds shape analysis. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021. p. 915–924.

XIANG, Y.; FOX, D. Da-rnn: Semantic mapping with data associated recurrent neural networks. In: *Robotics: Science and Systems (RSS)*. 2017.

XIANG, Y. et al. *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes*. : arXiv, 2017.

YAVARTANOO, M.; KIM, E. Y.; LEE, K. M. Spnet: Deep 3d object classification and retrieval using stereographic projection. In: JAWAHAR, C. et al. (Ed.). *Computer Vision – ACCV 2018*. Cham: Springer International Publishing, 2019. p. 691–706. ISBN 978-3-030-20873-8.

YUKSEL, C. Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum*, v. 34, 05 2015.

ZAPATA-IMPATA, B. S. et al. Fast geometry-based computation of grasping points on three-dimensional point clouds. *International Journal of Advanced Robotic Systems*, v. 16, n. 1, p. 1729881419831846, 2019.

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.